An algorithm for the rapid numerical evaluation of Bessel functions of real orders and arguments

James Bremer[a]

[a] *Department of Mathematics, University of California, Davis*

Abstract

We describe a method for the rapid numerical evaluation of the Bessel functions of the first and second kinds of nonnegative real orders and positive arguments. Our algorithm makes use of the well-known observation that although the Bessel functions themselves are expensive to represent via piecewise polynomial expansions, the logarithms of certain solutions of Bessel's equation are not. We exploit this observation by numerically precomputing the logarithms of carefully chosen Bessel functions and representing them with piecewise bivariate Chebyshev expansions. Our scheme is able to evaluate Bessel functions of orders between 0 and $1,000,000,000$ at essentially any positive real argument. In that regime, it is competitive with existing methods for the rapid evaluation of Bessel functions and has at least three advantages over them. First, our approach is quite general and can be readily applied to many other special functions which satisfy second order ordinary differential equations. Second, by calculating the logarithms of the Bessel functions rather than the Bessel functions themselves, we avoid many issues which arise from numerical overflow and underflow. Third, in the oscillatory regime, our algorithm calculates the values of a nonoscillatory phase function for Bessel's differential equation and its derivative. These quantities are useful for computing the zeros of Bessel functions, as well as for rapidly applying the Fourier-Bessel transform. The results of extensive numerical experiments demonstrating the efficacy of our algorithm are presented. A Fortran package which includes our code for evaluating the Bessel functions is publicly available.

Keywords: special functions, fast algorithms, nonoscillatory phase functions

1. Introduction

Here, we describe a numerical method for evaluating the Bessel functions of the first and second kinds — J_ν and Y_ν, respectively — of nonnegative orders and positive arguments. In this regime, it is competitive with (and possess some advantages over) existing methods for the numerical evaluation of the Bessel functions such as [1] and [22].

The purpose of this article, though, is not to argue that existing schemes for the evaluation of Bessel functions are inadequate or should be replaced with ours. Instead, it is to point out that there is an incredibly straightforward approach to their numerical evaluation that applies to a large class of

Email address: bremer@math.ucdavis.edu (James Bremer)

special functions satisfying second order ordinary differential equations. Our decision to focus in the first instance on Bessel functions stems in large part from the existence of satisfactory numerical algorithms with which we can compare our approach. The results of applying this approach to other classes of special functions, such as the associated Legendre functions and prolate spheroidal wave functions, will be reported by the author at a later date.

It is well known that the scaled Bessel functions $J_\nu(t)\sqrt{t}$ and $Y_\nu(t)\sqrt{t}$ satisfy the second order linear ordinary differential equation

$$y''(t) + \left(1 - \frac{\nu^2 - \frac{1}{4}}{t^2}\right) y(t) = 0 \quad \text{for all} \quad 0 < t < \infty. \tag{1}$$

We will, by a slight abuse of terminology, refer to (1) as Bessel's differential equation. When $0 \leq \nu \leq \frac{1}{2}$, the coefficient of y in (1) is positive on the half-line $(0,\infty)$, whereas it is negative on the interval

$$\left(0, \sqrt{\nu^2 - \frac{1}{4}}\right) \tag{2}$$

and positive on

$$\left(\sqrt{\nu^2 - \frac{1}{4}}, \infty\right) \tag{3}$$

when $\nu > \frac{1}{2}$. It follows from this and standard WKB estimates (see, for instance, [11]) that solutions of (1) are oscillatory on $(0,\infty)$ when $0 \leq \nu \leq \frac{1}{2}$, whereas they behave roughly like increasing or decreasing exponentials on (2) and are oscillatory on (3) when $\nu > \frac{1}{2}$. We will refer to the subset

$$\mathcal{O} = \left\{(\nu, t) : 0 \leq \nu \leq \frac{1}{2} \ \text{ and } t > 0\right\} \bigcup \left\{(\nu, t) : \nu > \frac{1}{2} \ \text{ and } t \geq \sqrt{\nu^2 - \frac{1}{4}}\right\} \tag{4}$$

of $\mathbb{R} \times \mathbb{R}$ as the oscillatory region and the subset

$$\mathcal{N} = \left\{(\nu, t) : \nu > \frac{1}{2} \ \text{ and } \ 0 < t < \sqrt{\nu^2 - \frac{1}{4}}\right\} \tag{5}$$

of $\mathbb{R} \times \mathbb{R}$ as the nonoscillatory region.

When ν is large, we cannot expect to represent Bessel functions efficiently using polynomial expansions in $\mathcal{N}$ because, in this event, they behave like rapidly increasing or decreasing exponentials. Similarly, we cannot expect to represent Bessel functions efficiently with polynomial expansions on any large subset of $\mathcal{O}$ since they oscillate there. Despite this, the logarithms of Bessel functions can be represented efficiently via polynomial expansions on $\mathcal{N}$. Moreover, there is a certain solution of Bessel's differential equation whose logarithm is nonoscillatory and hence can be represented efficiently via polynomial expansions on large subsets of the oscillatory region $\mathcal{O}$. This latter observation is related to the well-known fact that Bessel's differential equation admits a nonoscillatory phase function (see, for instance, Section 13.75 of [28] or [14]).

Many special functions of interest share this property of Bessel functions, at least in an asymptotic sense [24, 9]. However, the sheer effectiveness with which nonoscillatory phase functions can represent solutions of the general equation

$$y''(t) + \lambda^2 q(t) y(t) = 0 \quad \text{for all} \quad a < t < b \tag{6}$$

in which the coefficient q is smooth and positive appears to have been overlooked. Indeed, under mild conditions on q, it is shown in [5] that there exist a positive real number μ, a nonoscillatory function α and a basis of solutions $\{u, v\}$ of (6) such that

$$u(t) = \frac{\cos(\alpha(t))}{\sqrt{|\alpha'(t)|}} + O\left(\exp(-\mu\lambda)\right) \tag{7}$$

and

$$v(t) = \frac{\sin(\alpha(t))}{\sqrt{|\alpha'(t)|}} + O\left(\exp(-\mu\lambda)\right). \tag{8}$$

The constant μ is a measure of the extent to which q oscillates, with larger values of μ corresponding to greater smoothness on the part of q. The function α is nonoscillatory in the sense that it can be represented using various series expansions the number of terms in which do not vary with λ. That is, $O(\exp(-\mu\lambda))$ accuracy is obtained using an $O(1)$-term expansion. The results of [5] are akin to standard results on WKB approximation in that they apply to the more general case in which q varies with the parameter λ assuming only that q satisfies certain innocuous hypotheses independent of λ. A fast and highly accurate numerical algorithm for the computation of nonoscillatory phase functions for equations of the form (6) is described in [4], although we will not need it here since an effective asymptotic expansion of a nonoscillatory phase function for Bessel's differential equation is available. The algorithm of [4] is, however, of use in generalizing these results to cases in which such expansions are not available.

The algorithm of this paper operates by numerically precomputing the logarithms of certain solutions of Bessel's differential equation. We note that the nonoscillatory phase function is the imaginary part of the logarithm of a solution of Bessel's differential equation. Since there is a simple relationship between it and the real part (see the preliminaries, below), we store only the phase function when in the oscillatory regime. We represent these functions via piecewise bivariate Chebyshev expansions, the coefficients of which are arranged in a table. The table is, of course, stored on the disk and loaded into memory when needed so it only needs to be computed once. We supplement these precomputed expansions with asymptotic and series expansions in order to evaluate J_ν and Y_ν for all nonnegative real orders ν and positive real arguments. We note, though, that in cases in which such expansions are not available, the range of the parameter and argument covered by the precomputed expansions is sufficient for most purposes and could be extended if needed. The size of the precomputed table used in the experiments of this paper is roughly 1.3 megabytes.

The remainder of this paper is structured as follows. In Section 2, we review certain mathematical facts and numerical procedures which are used in the rest of this article. Section 3 details the operation of a solver for nonlinear differential equations which is used by the algorithm of Section 4 for the rapid solution of Bessel's differential equation (1) in the case in which the parameter ν is fixed. This procedure is, in turn, a component of the scheme for the construction of the precomputed table which we use to evaluate Bessel functions. That scheme is described in Section 5. Section 6 details our algorithm for the numerical evaluation of Bessel functions using this table and certain asymptotic and series expansions. Section 7 describes extensive numerical experiments performed in order to verify the efficacy of the algorithm of Section 6. We conclude with a few remarks regarding the contents of this article and possible directions for future work in Section 8.

2. Mathematical and Numerical Preliminaries

2.1. The condition number of the evaluation of a function

The condition number of the evaluation of a differentiable function $f : \mathbb{R} \to \mathbb{R}$ at the point x is commonly defined to be

$$\kappa_f(x) = \left| \frac{x f'(x)}{f(x)} \right| \tag{9}$$

(see, for instance, Section 1.6 of [15]). This quantity measures the ratio of the magnitude of the relative change in $f(x)$ induced by a small change in the argument x to the magnitude of the relative change in x in the sense that

$$\left| \frac{f(x+\delta) - f(x)}{f(x)} \right| \approx \kappa_f(x) \, \left| \frac{\delta}{x} \right| \tag{10}$$

for small δ. Since almost all quantities which arise in the course of numerical calculations are subject to perturbations with relative magnitudes on the order of machine epsilon, we consider

$$\kappa_f(x) \epsilon_0, \tag{11}$$

where ϵ_0 denotes machine epsilon, to be a rough estimate of the relative accuracy one should expect when evaluating $f(x)$ numerically (in fact, it tends to be a slightly pessimistic estimate). In the rest of this paper, we take ϵ_0 to be

$$\epsilon_0 = 2^{-52} \approx 2.22044604925031 \times 10^{-16}. \tag{12}$$

It is immediately clear from (9) that when $f'(x_0)x_0 \neq 0$ and $f(x_0) = 0$, $\kappa_f(x)$ diverges to ∞ as $x \to x_0$. One consequence of this is that there is often a significant loss of relative accuracy when $f(x)$ is evaluated near one of its roots. In order to avoid this issue, we will arrange for the functions we use to represent solutions of Bessel's equation to be bounded away from 0.

2.2. Series expansions of the Bessel functions

For complex-valued ν and $x > 0$, the Bessel function of the first kind of order ν is given by

$$J_\nu(x) = \sum_{j=0}^{\infty} \frac{(-1)^j}{\Gamma(j+1)\Gamma(j+\nu+1)} \left(\frac{x}{2}\right)^{2j+\nu}. \tag{13}$$

Here, we use the convention that

$$\frac{1}{\Gamma(j)} = 0 \tag{14}$$

whenever j is zero or a negative integer. Among other things, this ensures that (13) is still sensible when ν is a negative integer. When $x > 0$ and ν is not an integer, the Bessel function of the second kind of order ν is given by

$$Y_\nu(x) = \frac{\cos(\nu\pi) J_\nu(x) - J_{-\nu}(x)}{\sin(\nu\pi)}. \tag{15}$$

For integer values of ν, (15) loses its meaning; however, taking the limit of $Y_\nu(x)$ as $\nu \to n \in \mathbb{Z}$ yields

$$\begin{aligned} Y_n(x) = \frac{2}{\pi} J_n(x) \log\left(\frac{x}{2}\right) - \frac{1}{\pi} \sum_{j=0}^{n-1} \frac{\Gamma(n-j-1)}{\Gamma(j+1)} \left(\frac{x}{2}\right)^{2j-n} \\ - \frac{1}{\pi} \sum_{j=0}^{\infty} (-1)^j \, \frac{\psi(n+j+1) + \psi(j+1)}{\Gamma(j+1)\Gamma(n+j+1)} \left(\frac{x}{2}\right)^{n+2j}, \end{aligned} \tag{16}$$

where ψ is the logarithmic derivative of the gamma function. A derivation of this formula can be found in Section 7.2.4 of [10].

For the most part, when ν and x are of small magnitude, the value of $J_\nu(x)$ can be computed in a numerically stable fashion by truncating the series (13). In some cases, however, this can lead to numerical underflow. Accordingly, we generally evaluate the logarithm of J_ν by truncating the series in the expression

$$\log(J_\nu(x)) = -\log(\Gamma(\nu+1)) + \nu \log\left(\frac{x}{2}\right) + \log\left(\sum_{j=0}^{\infty} \frac{(-1)^j \Gamma(\nu+1)}{\Gamma(j+1)\Gamma(j+\nu+1)} \left(\frac{x}{2}\right)^j\right) \tag{17}$$

instead. Of course, this expression is only valid in the nonoscillatory regime, where $J_\nu(x)$ is positive.

On the other hand, Formula (15) can lead to significant errors when it is used to evaluate Y_ν numerically. In particular, when ν is close to, but still distinct from an integer, the evaluation of Y_ν via (15) results in significant round-off error due to numerical cancellation. Since Y_ν is analytic as a function of ν, this problem can be obviated by evaluating Y_ν via interpolation with respect to the order ν. Similarly, it is often more convenient to compute Y_ν when ν is an integer using interpolation than to do so via (16). Similar suggestions are made in [22].

The naive use of (15) can also lead to numerical overflow when ν is not close to an integer. In such cases we evaluate $\log(-Y_\nu(t))$ via

$$\log(-Y_\nu(x)) = \log\left(J_\nu(x)\right) + \log\left(\frac{-\cos(\nu\pi) + \exp(\log(J_{-\nu}(x)) - \log(J_\nu(x)))}{\sin(\nu\pi)}\right). \tag{18}$$

We calculate the logarithms of J_ν appearing in (18) using (17), of course.

2.3. Debye's asymptotic expansion for small arguments

The following form of Debye's asymptotic expansions can be found in [22]. For $x < \nu$ and N a nonnegative integer,

$$J_\nu(x) = \frac{1}{1 + \theta_{N+1,1}(\nu, 0)} \frac{\exp(-\eta)}{\sqrt{2\pi}(\nu^2 - x^2)^{\frac{1}{4}}} \times \left(\sum_{j=0}^{N} \frac{u_j(p)}{\nu^j} + \theta_{N+1,1}(\nu, p)\right) \tag{19}$$

and

$$Y_\nu(x) = -\sqrt{\frac{2}{\pi}} \frac{\exp(\eta)}{(\nu^2 - x^2)^{\frac{1}{4}}} \times \left(\sum_{j=0}^{N} (-1)^j \frac{u_j(p)}{\nu^j} + \theta_{N+1,2}(\nu, p)\right), \tag{20}$$

where

$$\eta = \nu \log\left(\frac{\nu}{x} + \sqrt{\left(\frac{\nu}{x}\right)^2 - 1}\right) - \sqrt{\nu^2 - x^2}, \tag{21}$$

$$p = \frac{\nu}{\sqrt{\nu^2 - x^2}}, \tag{22}$$

$\theta_{N+1,1}$ and $\theta_{N+1,2}$ are error terms, and u_0, u_1, ... are the polynomials defined via

$$u_0(t) = 1, \tag{23}$$

and the recurrence relation

$$u_{n+1}(t) = \frac{1}{2}\left(t^2 - t^4\right)\frac{du_n(t)}{dt} + \frac{1}{8}\int_0^t (1 - 5\tau^2) u_n(\tau)\, d\tau \quad \text{for all} \quad n \geq 0. \tag{24}$$

In [22], it is shown that there exist positive real constants $C_1, C_2, \ldots$ such that

$$\max\left\{\left|\theta_{N+1,1}(\nu,p)\right|, \left|\theta_{N+1,2}(\nu,p)\right|\right\} \leq 2\exp\left(\frac{2}{3g^{\frac{3}{2}}}\right)\frac{C_{N+1}}{g^{\frac{3}{2}(N+1)}}, \tag{25}$$

where

$$g = \frac{\nu - x}{\nu^{\frac{1}{3}}}, \tag{26}$$

for all $N \geq 0$. In other words, Debye's asymptotic expansions for small values of the parameter are uniform asymptotic expansions in inverse powers of the variable (26). See [22] for a further discussion of the implications of this observation.

The naive use of (19) and (20) when $t \ll \nu$ often results in numerical underflow and overflow. In order to avoid such problems, in this regime we evaluate the logarithms of the Bessel functions via the approximations

$$\log\left(J_\nu(x)\right) \approx -\eta - \frac{1}{4}\log(\nu^2 - x^2) + \log\left(\frac{1}{\sqrt{2\pi}}\sum_{j=0}^{N}\frac{u_j(p)}{\nu^j}\right) \tag{27}$$

and

$$\log(-Y_\nu(x)) \approx \eta - \frac{1}{4}\log(\nu^2 - x^2) + \log\left(\sqrt{\frac{2}{\pi}}\sum_{j=0}^{N}(-1)^j\frac{u_j(p)}{\nu^j}\right) \tag{28}$$

rather than evaluate the Bessel functions themselves.

Debye's asymptotic expansions are somewhat less efficient than the other methods used to evalate Bessel functions in this work. As a consequence, we prefer other approaches whenever possible.

2.4. The Riccati equation, Kummer's equation and phase functions

If $y = \exp(r(t))$ satisfies

$$y''(t) + q(t)y(t) = 0 \quad \text{for all} \quad t \in I, \tag{29}$$

where $I \subset \mathbb{R}$ is an open interval, then a straightforward computation shows that

$$r''(t) + (r(t))^2 + q(t) = 0 \quad \text{for all} \quad t \in I. \tag{30}$$

Equation (30) is known as the Riccati equation; an extensive discussion of it can be found, for instance, in [16]. By assuming that q is real-valued, and that

$$r(t) = \alpha(t) + i\beta(t) \tag{31}$$

with α and β real-valued, we obtain from (30) the system of ordinary differential equations

$$\begin{cases} \beta''(t) + (\beta'(t))^2 - (\alpha'(t))^2 + q(t) = 0 \\ \alpha''(t) + 2\alpha'(t)\beta'(t) = 0. \end{cases} \tag{32}$$

If α' is nonzero, then the second of these equations readily implies that

$$\beta(t) = -\frac{1}{2}\log\left(\left|\alpha'(t)\right|\right). \tag{33}$$

Inserting (33) into the first equation in (32) yields

$$q(t) - (\alpha'(t))^2 - \frac{1}{2}\left(\frac{\alpha'''(t)}{\alpha'(t)}\right) + \frac{3}{4}\left(\frac{\alpha''(t)}{\alpha'(t)}\right)^2 = 0. \tag{34}$$

We will refer to (34) as Kummer's equation after E. E. Kummer who studied it in [18]. We conclude that if the derivative of the function α is nonzero and satisfies (34), then

$$u(t) = \frac{\cos(\alpha(t))}{\sqrt{|\alpha'(t)|}} \tag{35}$$

and

$$v(t) = \frac{\sin(\alpha(t))}{\sqrt{|\alpha'(t)|}} \tag{36}$$

are solutions of the differential equation (29). A straightforward computation shows that the Wronskian of $\{u, v\}$ is 1, so that they form a basis in the space of solutions of this differential equation. In this event, the function α is said to be a phase function for (29).

Suppose, on the other hand, that $\tilde{u}$ and $\tilde{v}$ are real-valued solutions of (29), that the Wronskian of $\{\tilde{u}, \tilde{v}\}$ is 1, and that α is a smooth function such that

$$\alpha'(t) = \frac{1}{\tilde{u}(t)^2 + \tilde{v}(t)^2}. \tag{37}$$

Since $\tilde{u}$ and $\tilde{v}$ cannot simultaneously vanish on I, the expression on the denominator of (37) is never 0. A tedious (but straightforward) computation shows that (37) satisfies Kummer's equation, so that α is a phase function for (29) and the functions u, v defined via (35) and (36) form a basis in its space of solutions. We note, though, that since (37) only determines α up to a constant, u need not coincide with $\tilde{u}$ and v need not coincide with $\tilde{v}$.

2.5. A nonoscillatory phase function for Bessel's equation

In the case of the solutions

$$u_\nu(t) = \sqrt{\frac{\pi t}{2}} J_\nu(t) \tag{38}$$

and

$$v_\nu(t) = \sqrt{\frac{\pi t}{2}} Y_\nu(t) \tag{39}$$

of Bessel's differential equation, (37) becomes

$$\alpha_\nu'(t) = \frac{2}{\pi t}\ \frac{1}{J_\nu^2(t) + Y_\nu^2(t)}. \tag{40}$$

Note that the Wronskian of the pair $\{u_\nu, v_\nu\}$ is 1 on the interval $(0,\infty)$ (see, for instance, Formula (28) in Section 7.11 of [10]). We define a phase function α_ν for (1) via the formula

$$\alpha_\nu(t) = C + \int_0^t \alpha_\nu'(s)\ ds \tag{41}$$

with the constant C to be set so that

$$\frac{\cos(\alpha_\nu(t))}{\sqrt{\alpha_\nu'(t)}} = u_\nu(t) \tag{42}$$

and

$$\frac{\sin(\alpha_\nu(t))}{\sqrt{\alpha_\nu'(t)}} = v_\nu(t). \tag{43}$$

From (40) and the series expansions for J_ν and Y_ν appearing in Section 2.2, we see that

$$\lim_{t\to 0^+} \sqrt{\alpha_\nu'(t)}\ u_\nu(t) = 0 \tag{44}$$

while

$$\lim_{t\to 0^+} \sqrt{\alpha_\nu'(t)}\ v_\nu(t) = -1. \tag{45}$$

It follows that in order for

$$\lim_{t\to 0^+} \left(\frac{\cos(\alpha_\nu(t))}{\sqrt{\alpha_\nu'(t)}} - u_\nu(t)\right) = 0 = \lim_{t\to 0^+} \left(\frac{\sin(\alpha_\nu(t))}{\sqrt{\alpha_\nu'(t)}} - v_\nu(t)\right) \tag{46}$$

to hold, we must have

$$\cos(C) = \cos(\alpha_\nu(0)) = 0 \tag{47}$$

and

$$\sin(C) = \sin(\alpha_\nu(0)) = -1. \tag{48}$$

We conclude that by taking $C = -\pi/2$ — so that

$$\alpha_\nu(t) = -\frac{\pi}{2} + \int_0^t \alpha_\nu'(s)\ ds \tag{49}$$

— we ensure that (42) and (43) are satisfied.

Now we denote by M_ν the function appearing in denominator of the second factor in (40); that is, M_ν is defined via

$$M_\nu(t) = (J_\nu(t))^2 + (Y_\nu(t))^2. \tag{50}$$

A cursory examination of Nicholson's formula

$$M_\nu(t) = \frac{8}{\pi^2}\int_0^\infty K_0(2t\sinh(s))\cosh(2\nu s)\ ds \ \text{ for all } \ t > 0, \tag{51}$$

a derivation of which can be found in Section 13.73 of [28], shows that M_ν, and hence also α_ν' and α_ν, is nonoscillatory as a function of t. Figure 1 shows plots of the nonoscillatory functions α_ν and α_ν' when $\nu = 100$. We note that while α_ν and α_ν' can be represented efficiently via polynomial expansions in the oscillatory regime, it is clear from these plots that that is not the case in the nonoscillatory regime. The asymptotic expansion

$$M_\nu(t) \sim \frac{2}{\pi t} \sum_{n=0}^{\infty} \frac{\Gamma\left(n+\frac{1}{2}\right)}{\sqrt{\pi}\,\Gamma(n+1)} \frac{\Gamma\left(\nu+n+\frac{1}{2}\right)}{\Gamma\left(\nu-n+\frac{1}{2}\right)} \frac{1}{t^{2n}} \quad \text{as} \quad t \to \infty \tag{52}$$

can be derived easily from (51) (see Section 13.75 of [28]). The first few terms are

$$M_\nu \sim \frac{2}{\pi t}\left(1+\frac{1}{2}\frac{\mu-1}{(2t)^2}+\frac{1}{2}\cdot\frac{3}{4}\frac{(\mu-1)(\mu-9)}{(2t)^4}+\frac{1}{2}\cdot\frac{3}{4}\cdot\frac{5}{6}\frac{(\mu-1)(\mu-9)(\mu-25)}{(2t)^6}+\cdots\right), \tag{53}$$

where $\mu = 4\nu^2$.

From (52), we can derive an asymptotic expansion of (40). Indeed, if we denote the n^{th} coefficient in the sum appearing in (52) by r_n, then the coefficients $s_0, s_1, \ldots$ in the asymptotic expansion

$$\alpha_\nu'(t) \sim \sum_{n=0}^{\infty} \frac{s_n}{t^{2n}} \quad \text{as} \quad t \to \infty \tag{54}$$

are given by

$$s_0 = 1 \quad \text{and} \quad s_n = -\sum_{j=1}^{n} s_{n-j} r_j \quad \text{for all} \quad n \geq 1. \tag{55}$$

The proof of this is an exercise in elementary calculus, and can be found, for example, in Chapter 1 of [25]. We note that it follows from (52) that $r_0 = 1$, and that the coefficients $r_1, r_2 \ldots$ satisfy the recurrence relation

$$r_n = r_{n-1}\left(\frac{\mu-(2n-1)^2}{4}\right)\frac{2n-1}{2n}. \tag{56}$$

Using (55) and (56), as many terms as desired in the expansion (54) can be calculated either numerically or analytically. The first few are

$$\alpha_\nu'(t) \sim 1 - \frac{\mu-1}{8t^2} - \frac{\mu^2-26\mu+25}{128t^4} - \frac{\mu^3-115\mu^2+1187\mu-1073}{1024t^6} + \cdots. \tag{57}$$

Obviously, the indefinite integral of (57) is

$$\alpha_\nu(t) \sim \tilde{C} + t + \frac{\mu-1}{8t} + \frac{\mu^2-26\mu+25}{384t^3} + \frac{\mu^3-115\mu^2+1187\mu-1073}{5120t^5} + \cdots \tag{58}$$

with $\tilde{C}$ a constant to be determined to ensure compatibility with the definition (49). From the

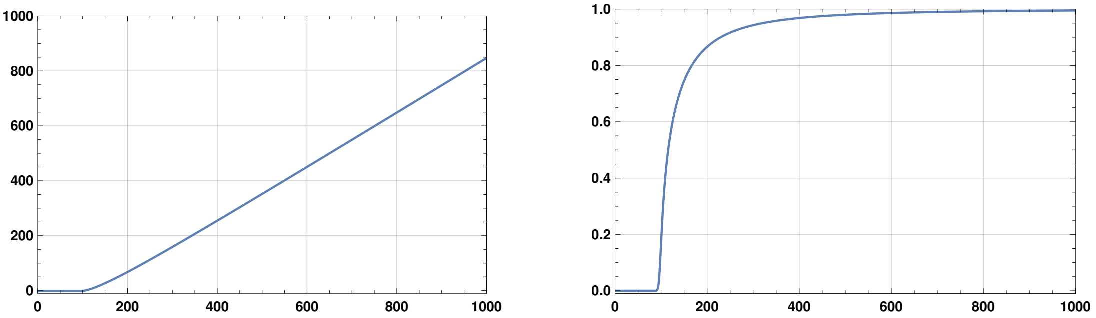


Figure 1: On the left is a plot of the phase function $\alpha_\nu(t)$ defined via (49) when $\nu = 100$, and on the right is a plot of its derivative with respect to t.

well-known asymptotic expansions

$$J_\nu(t) \sim \sqrt{\frac{2}{\pi t}} \cos\left(t - \frac{\pi}{2}\nu - \frac{\pi}{4}\right) \quad \text{as} \quad t \to \infty \tag{59}$$

and

$$Y_\nu(t) \sim \sqrt{\frac{2}{\pi t}} \sin\left(t - \frac{\pi}{2}\nu - \frac{\pi}{4}\right) \quad \text{as} \quad t \to \infty, \tag{60}$$

which can be found in Section 7.21 of [28] (for example), we see that it must be the case that

$$\tilde{C} = -\frac{\pi}{4} - \frac{\pi}{2}\nu + 2\pi L \tag{61}$$

with L an arbitrary integer. In order to set the integer L, we evaluated α_ν numerically via (49) for many large values of t and ν and found that (58) coincides with α_ν when L is taken to be 0. That is,

$$\alpha_\nu(t) \sim -\frac{\pi}{4} - \frac{\pi}{2}\nu + t + \frac{\mu - 1}{8t} + \frac{\mu^2 - 26\mu + 25}{384t^3} + \frac{\mu^3 - 115\mu^2 + 1187\mu - 1073}{5120t^5} + \cdots \quad \text{as} \quad t \to \infty. \tag{62}$$

We note that (62) can be found as Formula 10.18.17 in [9].

By differentiating (57) we obtain

$$\alpha_\nu''(t) \sim \frac{\mu - 1}{4t^3} + \frac{\mu^2 - 26\mu + 25}{32t^5} + \frac{3\mu^3 - 345\mu^2 + 3561\mu - 3219}{512t^7} + \cdots \quad \text{as} \quad t \to \infty. \tag{63}$$

In many instance, differentiating both sides of an asymptotic expansion such as (57) will not yield a valid expression. However, in this case it is permissible because α_ν' is analytic as function of t (see, for instance, Chapter I of [25] for a discussion of this issue).

While the construction of α_ν presented here is highly specialized to the case of Bessel's differential equation, the existence of nonoscillatory phase functions is an extremely general phenomenon. See [5] for a proof that under mild conditions on the coefficient q the differential equation

$$y''(t) + q(t)y(t) = 0 \quad \text{for all} \quad a < t < b \tag{64}$$

admits one.

2.6. Univariate Chebyshev series expansions

For $-1 \leq x \leq 1$ and integers $k \geq 0$, the Chebyshev polynomial T_k of degree k is given by the formula

$$T_k(x) = \cos(k \arccos(x)). \tag{65}$$

The Chebyshev series of a continuous function $f : [-1, 1] \to \mathbb{R}$ is

$$\sum_{j=0}^{\infty}{}' a_j T_j(x), \tag{66}$$

where the coefficients $a_0, a_1, \ldots$ are defined via

$$a_j = \frac{2}{\pi} \int_{-1}^{1} f(x) T_j(x) \frac{dx}{\sqrt{1 - x^2}} \tag{67}$$

and the dash next to the summation symbol indicates that the first term in the series is halved. It is a consequence of the well-known relationship between Fourier and Chebyshev series (as described, for instance, in Chapter 5 of [21]) and the celebrated theorem of Carleson [8] that (66) converges to

f pointwise almost everywhere on $[-1,1]$. Similarly, well-known results regarding the convergence of Fourier series imply that under mild smoothness assumptions on f, (66) converges uniformly to f on $[-1,1]$. See Theorem 5.7 in [21], which asserts that this is the case when f is of bounded variation, for an example of a result of this type.

If $f : [-1,1] \to \mathbb{R}$ can be analytically continued to the set

$$E_r = \left\{ z : \left| z + \sqrt{z^2+1} \right| < r \right\}, \tag{68}$$

where $r > 1$, then the rate of convergence of (66) can be estimated as follows:

$$\sup_{x\in[-1,1]} \left| \sum_{j=0}^{N}{}' a_j T_j(x) - f(x) \right| = \mathcal{O}\left(r^{-N}\right) \quad \text{as} \quad N \to \infty. \tag{69}$$

This result can be found as Theorem 5.16 in [21], among many other sources.

2.7. Bivariate Chebyshev series expansions

The bivariate Chebyshev series of a continuous function $f : [-1,1] \times [-1,1] \to \mathbb{R}$ is

$$\sum_{i=0}^{\infty}{}' \sum_{j=0}^{\infty}{}' a_{ij} T_i(x) T_j(y), \tag{70}$$

where the coefficients are defined via the formula

$$a_{ij} = \frac{4}{\pi^2} \int_{-1}^{1} \int_{-1}^{1} f(x,y) T_i(x) T_j(y) \frac{dx}{\sqrt{1-x^2}} \frac{dy}{\sqrt{1-y^2}} \tag{71}$$

and the dashes next to the summation symbols indicate that the first term in each sum is halved. It is an immediate consequence of the results of [12] on the pointwise almost everywhere convergence of multiple Fourier series that

$$\lim_{N\to\infty} \sum_{i=0}^{N}{}' \sum_{j=0}^{N}{}' a_{ij} T_i(x) T_j(y) = f(x,y) \tag{72}$$

for almost all $(x,y) \in [-1,1] \times [-1,1]$. As in the case of univariate Chebyshev series, under mild smoothness conditions on f, the convergence of (72) is uniform. See, for instance, Theorem 5.9 in [21].

The result of the preceding section on the convergence of the Chebyshev series of analytic functions can be generalized to bivariate Chebyshev expansions. In particular, if $f(x,y)$ is analytic on the set

$$\left\{ (x,y) \in \mathbb{C} \times \mathbb{C} : \ \left| x + \sqrt{x^2-1} \right| < r_1, \ \ \left| y + \sqrt{y^2-1} \right| < r_2 \right\}, \tag{73}$$

where $r_1, r_2 > 1$, then

$$|a_{ij}| = O\left(r_1^{-i} r_2^{-j} \right) \tag{74}$$

according to Theorem 11 in Chapter V of [2]. As a consequence, the Chebyshev series of f converges rapidly when f can be extended to an analytic function.

2.8. Chebyshev interpolation and spectral integration

In practice it is, of course, not possible to compute all of the coefficients in the Chebyshev series expansions (66) or (70), or even to compute the first few coefficients in these series exactly. Standard results, however, show that these coefficients of such a series can be approximated to high accuracy assuming that they decay rapidly.

For each nonnegative integer n, we refer to the collection of points

$$\rho_{j,n} = -\cos\left(\frac{\pi j}{n}\right), \quad j = 0, 1, \ldots, n, \tag{75}$$

as the $(n+1)$-point Chebyshev grid on the interval $[-1, 1]$, and we call individual elements of this set Chebyshev nodes or points. One discrete version of the well-known orthogonality relation

$$\int_{-1}^{1} \frac{T_i(x)T_j(x)}{\sqrt{1-x^2}}\, dx = \begin{cases} 0 & \text{if } \; i \neq j \\ \frac{\pi}{2} & \text{if } \; i = j > 0 \\ \pi & \text{if } \; i = j = 0. \end{cases} \tag{76}$$

is

$$\sum_{l\,=0}^{n}{}'' \, T_i(\rho_{l,n})T_j(\rho_{l,n}) = \begin{cases} 0 & \text{if } \; 0 \leq i, j \leq n \; \text{ and } i \neq j \\ \frac{n}{2} & \text{if } \; 0 < i = j < n \\ n & \text{if } \; i = j = 0 \; \text{ or } \; i = j = n. \end{cases} \tag{77}$$

Here, the double dash next to the summation sign indicates that the first and last term in the series are halved. Formula (77) can be found in a slightly different form in Chapter 4 of [21]. Any univariate polynomial f of degree n can be represented in the form

$$f(x) = \sum_{l=0}^{n}{}'' \, b_l T_l(x), \tag{78}$$

and the coefficients in the expansion (78) can be easily computed from the values of f at the nodes (75). In particular, it follows from (77) that

$$b_k = \frac{2}{n} \sum_{l=0}^{n}{}'' \, T_k\left(\rho_{l,n}\right) f\left(\rho_{l,n}\right) \tag{79}$$

for all $k = 0, 1, \ldots, n$.

If $f : [-1, 1] \to \mathbb{R}$ is smooth but no longer a polynomial, then the coefficients $b_0, b_1, \ldots, b_n$ obtained from (79) are related to the coefficients $a_0, a_1, \ldots$ in the Chebyshev expansion (66) of f via

$$b_k = a_k + \sum_{j=1}^{\infty} \left(a_{k+2jn} + a_{-k+2jn}\right) \quad \text{for all} \quad k = 0, 1, \ldots, n. \tag{80}$$

This result can be found in a slightly different form in Section 6.3.1 of [21]. It follows easily from (80) and the fact that the Chebyshev polynomials are bounded in $L^\infty\left([-1, 1]\right)$ norm by 1 that

$$\sup_{x \in [-1,1]} \left| f(x) - \sum_{l=0}^{n}{}'' \, b_l T_l(x) \right| \leq 2 \sum_{l=n+1}^{\infty} |a_l|. \tag{81}$$

In other words, assuming that the coefficients of the Chebyshev expansion of f decay rapidly, the series (78) converges rapidly to f as $n \to \infty$. We will, by a slight abuse of terminology, refer to (78) as the n^{th} order Chebyshev expansion of the function $f : [-1, 1] \to \mathbb{R}$.

Given only the values of a function $f : [-1, 1] \to \mathbb{R}$ at the nodes of the $(n+1)$-point Chebyshev grid on $[-1,1]$, it is possible to evaluate the Chebyshev expansion (78) in an efficient and numerical stable fashion without explicitly computing the coefficients $b_0, \ldots, b_n$. In particular, the value of (78) at a point $x \in [-1,1]$ which does not coincide with any of the grid points $\rho_{0,n}, \ldots, \rho_{n,n}$ is given by the barycentric interpolation formula

$$\left(\sum_{j=0}^{n} \frac{(-1)^j f(\rho_{j,n})}{x - \rho_{j,n}} \right) \Bigg/ \left(\sum_{j=0}^{n} \frac{(-1)^j}{x - \rho_{j,n}} \right). \tag{82}$$

See, for instance, [27] for a thorough discussion of the numerical stability and efficiency of this technique.

For each $k > 1$, the general antiderivative of T_k is

$$\int T_k(x)\ dx = \frac{1}{2} \left(\frac{T_{k+1}(x)}{k+1} - \frac{T_{k-1}(x)}{k-1} \right) + C \tag{83}$$

while

$$\int T_0(x)\ dx = \frac{1}{4} T_1(x) + C \tag{84}$$

and

$$\int T_1(x)\ dx = \frac{1}{4} T_2(x) + C. \tag{85}$$

These formulas can be found, for instance, in Section 2.4.4 of [21]. Using them, the values of

$$g(x) = \int_{-1}^{x} f(t)\ dt, \tag{86}$$

where f is defined via (78), can be computed at the Chebyshev nodes (75). We will refer to the matrix which takes the values of f at the nodes (78) to those of g at the same nodes as the $(n+1) \times (n+1)$ spectral integration matrix.

There results for univariate Chebyshev expansions can be easily generalized to the case of bivariate Chebyshev expansions. Given $f : [-1,1] \times [-1,1] \to \mathbb{R}$, we will refer to the series

$$\sum_{i=0}^{n}{}'' \sum_{j=0}^{n}{}'' b_{ij} T_i(x) T_j(y) \tag{87}$$

whose the coefficients $\{b_{ij} : 0 \leq i,j \leq n\}$ are defined via the formula

$$b_{ij} = \frac{4}{n^2} \sum_{l=0}^{n}{}'' \sum_{k=0}^{n}{}'' T_i\left(\rho_{l,n}\right) T_j\left(\rho_{k,n}\right) f\left(\rho_{l,n}, \rho_{k,n}\right) \tag{88}$$

as the n^{th} order Chebyshev expansion of f. It is easy to see that

$$\sup_{x \in [-1,1]} \left| f(x,y) - \sum_{i=0}^{n}{}'' \sum_{j=0}^{n}{}'' b_{ij} T_i(x) T_j(y) \right| \leq 2 \sum_{i=n+1}^{\infty} \sum_{j=n+1}^{\infty} |a_{ij}|, \tag{89}$$

where the $\{a_{ij}\}$ are defined via (71).

2.9. Compressed bivariate Chebyshev expansions

It often happens that many of the coefficients in the bivariate Chebyshev expansion (87) of a function $f : [-1,1] \times [1,1] \to \mathbb{R}$ are of negligible magnitude. In order to reduce the cost of storing such expansions as well as the cost of evaluating them, we use the following construction to reduce the number of coefficients which need to be considered.

Suppose that $\epsilon > 0$, and that

$$\sum_{i=0}^{n}{}'' \sum_{j=0}^{n}{}'' b_{ij} T_i(x) T_j(y) \tag{90}$$

is the n^{th} order Chebyshev expansion for $f : [-1,1] \times [-1,1] \to \mathbb{R}$. We let M denote the least positive integer which is less than or equal to n and such that

$$|b_{ij}| < \epsilon \quad \text{for all} \quad i > M \quad \text{and} \quad j = 0, \ldots, n, \tag{91}$$

assuming such an integer exists. If it does not, then we take $M = n$. Similarly, for each $i = 0, \ldots, M$, we let m_i be the least positive integer less than or equal to n such that

$$|b_{ij}| < \epsilon \quad \text{for all} \quad j = m_i + 1, \ldots, n \tag{92}$$

if such an integer exists, and we let $m_i = n$ otherwise. We refer to the series

$$\sum_{i=0}^{M} \sum_{j=0}^{m_i} \widetilde{b_{ij}} T_i(x) T_j(x), \tag{93}$$

where $\widetilde{b_{ij}}$ is defined via

$$\widetilde{b_{ij}} = \begin{cases} b_{ij} & \text{if} \quad 1 \le i, j \le n \\ \frac{1}{2} b_{ij} & \text{if} \quad 1 \le j \le n \quad \text{and} \quad i = 0, \ n \\ \frac{1}{2} b_{ij} & \text{if} \quad 1 \le i \le n \quad \text{and} \quad j = 0, \ n \\ \frac{1}{4} b_{ij} & \text{otherwise,} \end{cases} \tag{94}$$

as the ϵ-compressed n^{th} order Chebyshev expansion of f.

Obviously, the results discussed in Sections 2.6 through Section 2.9 can be modified in a straightforward fashion so as to apply to a function given on an arbitrary interval $[a, b]$ (in the case of univariate functions) or one given on a compact rectangle $[a, b] \times [c, d]$ (in the case of a bivariate function). For instance, the nodes of the $(n + 1)$-point Chebyshev grid on $[a, b]$ are

$$\tilde{\rho}_{j,n} = -\frac{b-a}{2} \cos\left(\frac{\pi j}{n}\right) + \frac{b+a}{2}, \quad j = 0, 1, \ldots, n \tag{95}$$

and the n^{th} order Chebyshev expansion of $f : [a, b] \to \mathbb{R}$ is

$$\sum_{i=0}^{n}{}'' b_i T_i \left(\frac{2}{b-a} x + \frac{b+a}{a-b} \right), \tag{96}$$

where the coefficients $b_0, \ldots, b_n$ are given by

$$b_i = \frac{2}{n} \sum_{l=0}^{n}{}'' T_i \left(\rho_{l,n}\right) f\left(\tilde{\rho}_{l,n}\right). \tag{97}$$

2.10. An adaptive discretization procedure

We now briefly describe a fairly standard procedure for adaptively discretizing a smooth function $f : [a, b] \to \mathbb{R}$. It takes as input a desired precision $\epsilon > 0$ and a positive integer n. The goal of this procedure is to construct a partition

$$a = \gamma_0 < \gamma_1 < \cdots < \gamma_m = b \tag{98}$$

of $[a, b]$ such that the n^{th} order Chebyshev expansion of f on each of the subintervals $[\gamma_j, \gamma_{j+1}]$ of $[a, b]$ approximates f with accuracy ϵ. That is, for each $j = 0, \ldots, m-1$ we aim to achieve

$$\sup_{x\in[\gamma_j,\gamma_{j+1}]} \left| f(x) - \sum_{i=0}^{n}{}'' b_{i,j} T_i \left(\frac{2}{\gamma_{j+1} - \gamma_j} x + \frac{\gamma_{j+1} + \gamma_j}{\gamma_j - \gamma_{j+1}} \right) \right| < \epsilon, \tag{99}$$

where $b_{0,j}, b_{1,j} \ldots, b_{n,j}$ are the coefficients in the n^{th} order Chebyshev expansion of f on the interval $[\gamma_j, \gamma_{j+1}]$. These coefficients are defined by the formula

$$b_{i,j} = \frac{2}{n} \sum_{l=0}^{n}{}'' T_i\left(\rho_{l,n}\right) f \left(\frac{\gamma_j - \gamma_{j+1}}{2} \cos\left(\frac{\pi l}{n}\right) + \frac{\gamma_{j+1} + \gamma_j}{2} \right). \tag{100}$$

During the procedure, two lists of subintervals are maintained: a list of subintervals which are to be processed and a list of output subintervals. Initially, the list of subintervals to be processed consists of $[a, b]$ and the list of output subintervals is empty. The procedure terminates when the list of subintervals to be processed is empty or when the number of subintervals in this list exceeds a present limit (we usually take this limit to be 300). In the latter case, the procedure is deemed to have failed. As long as the list of subintervals to process is nonempty and its length does not exceed the preset maximum, the algorithm proceeds by removing a subinterval $[\eta_1, \eta_2]$ from that list and performing the following operations:

1. Compute the coefficients $b_0, \ldots, b_n$ in the n^{th} order Chebyshev expansion of the restriction of f to the interval $[\eta_1, \eta_2]$.

2. Compute the quantity
$$\Delta = \frac{\max\left\{\left|b_{\frac{n}{2}+1}\right|, \left|b_{\frac{n}{2}+2}\right|, \ldots \left|b_n\right|\right\}}{\max\left\{\left|b_0\right|, \left|b_1\right|, \ldots \left|b_n\right|\right\}}. \tag{101}$$

3. If $\Delta < \epsilon$ then the subinterval $[\eta_1, \eta_2]$ is added to the list of output subintervals.

4. If $\Delta \geq \epsilon$, then the subintervals
$$\left[\eta_1, \frac{\eta_1 + \eta_2}{2}\right] \quad \text{and} \quad \left[\frac{\eta_1 + \eta_2}{2}, \eta_2\right] \tag{102}$$
are added to the list of subintervals to be processed.

This algorithm is heuristic in the sense that there is no guarantee that (99) will be achieved, but similar adaptive discretization procedures are widely used with great success.

There is one common circumstance which leads to the failure of this procedure. The quantity Δ is an attempt to estimate the relative accuracy with which the Chebyshev expansion of f on the interval $[\eta_1, \eta_2]$ approximates f. In cases in which the condition number of the evaluation of f is larger than ϵ on some part of $[a, b]$, the procedure will generally fail or an excessive number of subintervals will be generated. Particular care needs to be taken when f has a zero in $[a, b]$. In most cases, for x near a zero of f, the condition number of evaluation of $f(x)$ (as defined in Section 2.1) is large. In this article, we avoid such difficulties by only applying this procedure to functions which are bounded away from 0.

3. An adaptive solver for nonlinear differential equations

In this section, we describe a numerical algorithm for the solution of nonlinear second order differential equations of the form

$$y''(t) = f(t, y(t), y'(t)) \quad \text{for all} \quad a < t < b. \tag{103}$$

It is intended to be extremely robust, but not necessarily highly efficient. This is fitting since we only use it to perform precomputations. Here, we assume that initial conditions for the desired solution y of (103) are specified. That is, we seek a solution of (103) which satisfies

$$y(a) = y_a \quad \text{and} \quad y'(a) = y'_a, \tag{104}$$

where the constants y_a and y'_a are given. The algorithm can easily be modified for the case of a terminal value problem.

The procedure takes as input a subroutine for evaluating f and its derivatives with respect to y and y', a positive integer n, and a positive real number $\epsilon > 0$. It maintains a list of output subintervals, a stack containing a set of subintervals to process, and two constants c_1 and c_2. Initially, the list of output subintervals is empty, the stack consists of $[a, b]$, c_1 is taken to be y_a, and c_2 is taken to be y'_a. If the size of the stack exceeds a preset maximum (taken to be 300 in the calculations performed in this paper), the procedure is deemed to have failed. As long as the stack is nonempty and its length does not exceed the preset maximum, the algorithm proceeds by popping an interval $[\eta_1, \eta_2]$ off of the stack and performing the following operations:

1. Form the nodes $t_0, t_1, \ldots, t_n$ of the $(n+1)$-point Chebyshev grid on the interval $[\eta_1, \eta_2]$.

2. Apply the trapezoidal method (see, for instance, [17]) in order to approximate the values of the second derivative y''_0 of the function satisfying

$$\begin{cases} y''_0(t) = f(t, y(t), y'(t)) \quad \text{for all} \quad \eta_1 \le t \le \eta_2 \\ y_0(\eta_1) = c_1 \\ y'_0(\eta_1) = c_2 \end{cases} \tag{105}$$

   at the points $t_0, t_1, \ldots, t_n$.

3. Using spectral integration, approximate the values of y'_0 and y_0 at the points $t_0, \ldots, t_n$ through the formulas

$$y'_0(t) = c_2 + \int_{\eta_1}^{t} y''_0(s)\, ds \tag{106}$$

   and

$$y_0(t) = c_1 + \int_{\eta_1}^{t} y'_0(s)\, ds. \tag{107}$$

4. Apply Newton's method to the initial value problem (105). The function y_0 is used as the initial guess for Newton's method. The j^{th} iteration of Newton's method starts with an initial

approximation y_{j-1} and consists of solving the linearized problem

$$\delta_j''(t) - \frac{\partial f}{\partial y}(t, y_0(t), y_0'(t))\delta_j(t) - \frac{\partial f}{\partial y'}(t, y_0(t), y_0'(t))\delta_j'(t) = f(t, y_0(t), y_0'(t)) - y_0''(t) \tag{108}$$

on the interval $[\eta_1, \eta_1]$ for δ_j and forming the new approximation $y_j = y_{j-1} + \delta_j$. The initial conditions

$$\delta_j(0) = \delta_j'(0) = 0 \tag{109}$$

are imposed since y_0 is already consistent with the desired initial conditions. All of the functions appearing in (108) are represented via their values at the points $t_0, t_1, \dots, t_n$. Of course, the values of a function at these nodes implicitly defines its n^th order Chebyshev expansion.

An integral equation method is used to solve (108). More specifically, by assuming that δ_j is given by

$$\delta_j(t) = \int_{\eta_1}^{t} \int_{\eta_1}^{s} \sigma_j(\tau)\ d\tau\ ds \tag{110}$$

the system (108) is transformed into the integral equation

$$\begin{aligned} \sigma_j(t) - \frac{\partial f}{\partial y}(t, y_{j-1}(t), y_{j-1}'(t)) \int_{\eta_1}^{t} \int_{\eta_1}^{s} \sigma_j(\tau)\ d\tau\ ds - \\ \frac{\partial f}{\partial y'}(t, y_{j-1}(t), y_{j-1}'(t)) \int_{\eta_1}^{t} \sigma_j(\tau)\ d\tau\ ds = f(t, y_{j-1}(t), y_{j-1}'(t)) - y_{j-1}''(t). \end{aligned} \tag{111}$$

We note that the choice of the representation (110) of δ_j is consistent with the conditions (109). The linear system which arises from requiring that (111) is satisfied at the points $t_0, t_1, \dots, t_n$ is inverted in order to calculate the values of σ_j at those nodes. Spectral integration (as described in Section 2.9) is used to evaluate the integrals, and to compute the values of δ at $t_0, \dots, t_n$ via (110).

Define

$$\Delta_j = \max\left\{|\delta_j(t_0)|, |\delta_j(t_1)|, \dots, |\delta_j(t_n)|\right\}. \tag{112}$$

If $j > 1$ and $\Delta_j < \Delta_{j-1}$, then Newton iterations continue. Otherwise, the Newton iteration is terminated, having obtained y_{j-1} as the result of the procedure.

5. Compute the Chebyshev coefficients $b_0, b_1, \dots, b_n$ of the polynomial which interpolate the values of y_{j-1} at the points $t_0, t_1, \dots, t_n$ and define Λ via

$$\Lambda = \frac{\max\left\{\left|b_{\frac{n}{2}+1}\right|, \left|b_{\frac{n}{2}+2}\right|, \dots |b_n|\right\}}{\max\left\{|b_1|, |b_2|, \dots |b_n|\right\}}. \tag{113}$$

If $\Lambda > \epsilon$, then push the subintervals

$$\left[\frac{\eta_1 + \eta_2}{2}, \eta_2\right] \tag{114}$$

and

$$\left[\eta_1, \frac{\eta_1 + \eta_2}{2}\right] \tag{115}$$

onto the stack (in that order) so that (115) is the next interval to be processed by the algorithm.

If $\Lambda \leq \epsilon$, then $[\eta_1, \eta_2]$ is added to the list of output intervals, c_1 is set equal to $y_j(\eta_2)$, and c_2 is set equal to $y_j'(\eta_2)$.

As in the case of the adaptive discretization procedure of Section 2.10, this is a heuristic algorithm which is not guaranteed to achieve an accurate discretization of the solution of (103). Moreover, the quantity Λ defined in (113) is an attempt to measure the relative accuracy with which the obtained solution of (103) is represented on the interval under consideration. When the condition number of evaluation of the solution y of (103) is large, this algorithm tends to produce an excessive number of intervals or fail altogether. Since the condition number of evaluation of a function f is generally large near its zeros, in this article we always apply it in cases in which the solution of (103) is bounded away from 0.

Remark 1. *When applied to (103), the trapezoidal method produces approximations of the values of y_0 and y_0' at the nodes $t_0, t_1, \ldots, t_n$ in addition to approximations of the values of y_0''. We discard those values and recompute y_0 and y_0' via spectral integration. We do so because while the values of y_0 and y_0' obtained by the trapezoidal method at $t_0, t_1, \ldots, t_n$ must satisfy the relations*

$$y_0''(t_j) = f(t_j, y(t_j), y'(t_j)) \quad \textit{for all} \quad j = 0, 1, \ldots, n,$$

they need not be consistent with each other in the sense that

$$\begin{pmatrix} y_0'(t_0) \\ y_0'(t_1) \\ \vdots \\ y_0'(t_n) \end{pmatrix} = \begin{pmatrix} c_2 \\ c_2 \\ \vdots \\ c_2 \end{pmatrix} + S_n \begin{pmatrix} y_0''(t_0) \\ y_0''(t_1) \\ \vdots \\ y_0''(t_n) \end{pmatrix} \tag{116}$$

and

$$\begin{pmatrix} y_0(t_0) \\ y_0(t_1) \\ \vdots \\ y_0(t_n) \end{pmatrix} = \begin{pmatrix} c_1 \\ c_1 \\ \vdots \\ c_1 \end{pmatrix} + S_n \begin{pmatrix} y_0'(t_0) \\ y_0'(t_1) \\ \vdots \\ y_0'(t_n) \end{pmatrix}, \tag{117}$$

where S_n denotes the $(n+1) \times (n+1)$ spectral integration matrix, might not hold. Proceeding without recomputing the values of y_0 and y_0' in order to make sure that these consistency conditions are satisfied would lead to the failure of Newton's method in most cases.

4. An algorithm for the rapid numerical solution of Bessel's differential equation

In this section, we describe a numerical algorithm for the solution of Bessel's differential equation for a fixed value of ν. Our algorithm runs in time independent of ν and is a key component of the scheme of the following section for the construction of tables which allow for the rapid numerical evaluation of the Bessel functions.

The algorithm takes as input $\nu \geq 0$ and a desired precision $\epsilon > 0$. It proceeds in three stages.

Stage one: computation of a nonoscillatory phase function

In this stage, we calculate the nonoscillatory phase function α_ν defined by (49) on the interval

$$\left[\sqrt{\nu^2-\frac{1}{4}},\ 1000\ \nu\right] \tag{118}$$

if $\nu > \frac{1}{2}$, and on the interval

$$[2, 1000] \tag{119}$$

in the event that $0 \leq \nu \leq \frac{1}{2}$. In either case, we will denote the left-hand side of the interval on which we calculate α_ν by a and the right-hand side by b.

We first construct the derivative α_ν' of α_ν by solving Kummer's equation (34) on the interval $[a, b]$ with with q taken to be the coefficient of y in Bessel's differential equation (1); that is,

$$q(t) = 1 - \frac{\nu^2 - \frac{1}{4}}{t^2}. \tag{120}$$

Most solutions of (34) are oscillatory; however, the phase function α_ν is a nonoscillatory. Moreover, the values of α_ν' and its derivative α_ν'' at the right-hand endpoint b can be approximated to high accuracy via the asymptotic expansions (54) and (63). Accordingly, we solve a terminal value problem for Kummer's equation with the values of $\alpha_\nu'(b)$ and $\alpha_\nu''(b)$ specified. We use the adaptive procedure of Section 3 to solve Kummer's equation; the input n to that procedure is taken to be 30 and the desired precision is set to ϵ. The functions α_ν' and α_ν'' are represented via their values at the nodes of the 31-point Chebyshev grids (see Section 2.8) on a collection of subintervals

$$[\gamma_0, \gamma_1], [\gamma_1, \gamma_2], \ldots, [\gamma_{m-1}, \gamma_m], \tag{121}$$

where $a = \gamma_0 < \gamma_1 < \ldots < \gamma_m = b$ is a partition of $[a, b]$ which is determined adaptively by the solver of Section 3. We use the formula

$$\alpha_\nu(t) = \alpha_\nu(b) + \int_b^t \alpha_\nu'(s)\ ds \tag{122}$$

to calculate α_ν. More specifically, spectral integration is used to obtain the values of α_ν at the nodes of the 31-point Chebyshev grids on the subintervals (121). The value of $\alpha_\nu(b)$ is approximated to high accuracy via the asymptotic expansion (62). We use the first 30 terms of each of the expansions (62), (54) and (63).

The functions α_ν, α_ν' and α_ν'' can calculated in an efficient and numerically stable fashion at any point in the interval $[a, b]$ via barycentric Chebyshev interpolation using their values at the nodes of the Chebyshev grids on the subintervals (121) (see Section 2.8). Using the values of α_ν and α_ν', J_ν and Y_ν can be evaluated at any point on the interval $[a, b]$ via (38) and (39).

Step two: computation of $\nu + \log\left(-Y_\nu(t)\sqrt{t}\right)$

In the event that $\nu > \frac{1}{2}$, we calculate the function $\nu + \log\left(-Y_\nu(t)\sqrt{t}\right)$ on the interval

$$\left[\frac{\nu}{1000}, \sqrt{\nu^2 - \frac{1}{2}}\right] \tag{123}$$

by solving a terminal value problem for Riccati's equation

$$r''(t) + (r'(t))^2 + q(t) = 0 \tag{124}$$

with q given by (120). In fact, we solve Riccati's equation on the slightly larger interval

$$\left[\frac{\nu}{1000}, t^*\right], \tag{125}$$

where t^* is the solution of the equation

$$\alpha_\nu\left(t^*\right) = \frac{\pi}{2}. \tag{126}$$

That there exists a solution t^* of this equation such that

$$t^* > \sqrt{\nu^2 - \frac{1}{4}} \tag{127}$$

is a consequence of a well-known result regarding the zeros of Bessel functions; namely, that J_ν cannot have zeros on the interval

$$\left(0, \sqrt{\nu^2 - \frac{1}{4}}\right] \tag{128}$$

(see, for instance, Chapter 15 of [28]). From (42), we see that the zeros of J_ν occur at points t such that

$$\alpha_\nu(t) = \frac{\pi}{2} + \pi k \quad \text{with} \quad k \in \mathbb{Z}. \tag{129}$$

It is obvious from the definition (49) of α_ν that $\alpha_\nu(0) = -\frac{\pi}{2}$, and that α_ν is increasing as a function of t. Consequently, if t^* denotes the smallest positive real number such that $J_\nu\left(t^*\right) = 0$, then t^* satisfies (126) and and it must be the case that

$$t^* > \sqrt{\nu^2 - \frac{1}{4}} \tag{130}$$

since there are no zeros of J_ν in (128). The values of α_ν and its derivative having been calculated in the preceding phase, there is no difficulty in using Newton's method to obtain the value of t^* by solving the nonlinear equation (126) numerically. Moreover, the values of the functions Y_ν and Y_ν' at t^* can be calculated without the loss of precision indicated by their condition numbers of evaluation (see Section 2.1 for a definition of the condition number of evaluation of a function). In particular, since $\alpha_\nu(t^*) = -\frac{\pi}{2}$,

$$Y_\nu(t^*) = \sqrt{\frac{2}{\pi t}} \frac{\sin(\alpha_\nu(t^*))}{\sqrt{\alpha_\nu'(t^*)}} = \sqrt{\frac{2}{\pi t \ \alpha_\nu'(t^*)}} \tag{131}$$

and

$$Y_\nu'(t^*) = \sqrt{\frac{2}{\pi t}} \cos(\alpha_\nu(t^*)) \sqrt{\alpha_\nu'(t^*)} - \sqrt{\frac{2}{\pi t}} \frac{\sin(\alpha_\nu(t^*))\alpha_\nu''(t^*)}{2(\alpha_\nu'(t^*))^{\frac{3}{2}}} = -\sqrt{\frac{1}{2\pi t}} \frac{\alpha_\nu''(t)}{(\alpha_\nu'(t))^{\frac{3}{2}}}. \tag{132}$$

The condition number of the evaluation of the nonoscillatory functions α_ν' and α_ν'' is not large and is bounded independent of ν, so there calculations can be performed without much loss of accuracy. See, for instance, [3], where this issue is discussed in detail. We note that the numerical evaluation of J_ν and Y_ν at an arbitrary point t via (42) and (43) will result in a relative error on the order of the condition number of the evaluation of these functions. This loss of accuracy stems from the evaluation of the trigonometric functions cosine and sine which appear in those formulas.

From the values of Y_ν and Y_ν' at t^*, we calculate the values of $\nu + \log(-Y_\nu(t)\sqrt{t})$ and its derivative there. Then we solve the corresponding terminal value problem for Riccati's equation. We use the solver described in Section 3 to do so. Our motivation for calculating $\nu + \log(-Y_\nu(t)\sqrt{t})$ in lieu of $\log(-Y_\nu(t)\sqrt{t})$ is that the former is bounded away from 0 on the interval (125) while the latter is not. As discussed in Section 2.1 the condition number of evaluation of a function near one of its roots is typically large and this causes difficulties for the adaptive solver of Section 3.

As with the phase function α_ν and its derivative, the function $\nu + \log(-Y_\nu(t)\sqrt{t})$ is represented via its values at the 31-point Chebyshev grid on a collection of subintervals of (125). It can be evaluated via barycentric Chebyshev interpolation at any point on that interval, and the values of Y_ν can obviously be obtained from those of $\nu + \log(-Y_\nu(t)\sqrt{t})$.

Stage three: computation of $-\nu + \log(J_\nu(t)\sqrt{t})$

Assuming that $\nu > \frac{1}{2}$, we calculate the function $-\nu + \log(J_\nu(t)\sqrt{t})$ on the interval

$$\left(0, \sqrt{\nu^2 - \frac{1}{4}}\right]. \tag{133}$$

This function is a solution of the Riccati equation (133) with q as in (120), and it is tempting to try to calculate in the same way that $\nu + \log(Y_\nu(t)\sqrt{t})$ is constructed in the preceding step. That is, by evaluating J_ν and its derivative at a suitably chosen point

$$t^{**} > \sqrt{\nu^2 - \frac{1}{4}} \tag{134}$$

and solving the corresponding terminal value problem for Riccati's equation. Such an approach is not numerically viable. The solution

$$-\nu + \log(J_\nu(t)\sqrt{t}) \tag{135}$$

is recessive when solving the Riccati equation in the backward direction while

$$\nu + \log(-Y_\nu(t)\sqrt{t}) \tag{136}$$

is dominant. As a consequence, approximations of (135) obtained by solving a terminal boundary value problem for (124) are highly inaccurate while approximations of (136) obtained in such a fashion are not. See, for instance, Chapter I of [13] for a discussion of the recessive and dominant solutions of ordinary differential equations.

Rather than solving a terminal boundary value for (124) in order to calculate (135), we solve an initial value problem. When $\nu \geq 10$, we use the logarithm form (27) of Debye's asymptotic of J_ν in order to evaluate (135) and its derivative at the left-hand endpoint of (133). When $\nu < 10$, Debye's expansion is not necessarily sufficiently accurate and we use the series expansion (17) in order to evaluate (135) and its derivative at the left-hand endpoint of (133). Again, our motivation for calculating $-\nu + \log(J_\nu(t)\sqrt{t})$ in lieu of $\log(J_\nu(t)\sqrt{t})$ is that the former is bounded away from 0 on the interval (133) while the latter is not.

The initial value problem is solved using the procedure of Section 3, and, as in the cases of α_ν and $\nu + \log(-Y_\nu(t)\sqrt{t})$, we represent $-\nu + \log(J_\nu(t)\sqrt{t})$ via its value at the 31-point Chebyshev grid on a collection of subintervals of (133). Using this data, the value of J_ν can be evaluated at any point in the interval $[a, b]$ via the obvious procedure.

Remark 2. *Although the algorithm described in this section is highly specialized to the case of*

Bessel's differential equation, it can, in fact, be modified so as to apply to a large class of second order equations of the form

$$y''(t) + q(t)y(t) = 0 \quad \textit{for all} \quad a < t < b. \tag{137}$$

Suppose, for instance, that q is smooth on $[a, b]$, has a zero at $t_0 \in (a, b)$, is negative on (a, t_0) and is positive on (t_0, b). The procedure of the first stage for constructing a nonoscillatory phase function on (t_0, b) relies on an asymptotic expansion which allows for the evaluation of a nonoscillatory phase function at the point b. In the absence of such an approximation, the algorithm of [4] can be used instead. That algorithm also proceeds by solving Kummer's equation (34), but it incorporates a mechanism for numerically calculating the appropriate initial values of a nonoscillatory phase function and its derivatives.

The procedure of the second stage does not rely on any asymptotic or series expansions of Bessel functions, only on the values of the phase function computed in the first phase. Consequently, it does not need to be modified in order to obtain a solution of Riccati's equation which is increasing as $t \to 0^+$.

In the third stage, one of Debye's asymptotic expansions is used to compute the values of the Bessel function J_ν and its derivative at a point near 0. In the event that such an approximation is not available, a solution of the Riccati equation which is increasing as $t \to t_0$ from the left can be obtained by solving an initial value problem with arbitrary initial conditions and then scaling the result in order to make it consistent with the desired solution of (137). This procedure is analogous to that used in order to obtain a recessive solution of a linear recurrence relation by running the recurrence relation backwards (see, for instance, Section 3.6 of [9]).

Further generalization to the case in which q has multiple zeros on the interval $[a, b]$ is also possible, but beyond the scope of this article.

5. The numerical construction of the precomputed table

In this section, we describe the procedure used to construct the table which allows for the numerical evaluation of the Bessel functions J_ν and Y_ν for a large range of parameters and arguments. This table stores the coefficients in the piecewise compressed bivariate Chebyshev expansions (as defined in Section 2.9) of several functions.

A first set of functions A_1 and C_1 allow for the evaluation of the nonoscillatory phase function $\alpha_\nu(t)$ defined in Section 2.5, as well as its derivative $\alpha_\nu'(t)$, on the subset

$$\mathcal{O}_1 = \left\{(\nu, t) : 2 \le \nu \le 1{,}000{,}000{,}000 \ \text{ and } \sqrt{\nu^2 - \frac{1}{4}} \le t \le 1000\nu\right\} \tag{138}$$

of the oscillatory region $\mathcal{O}$. A second set of functions A_2 and C_2 allow for the evaluation of $\alpha_\nu(t)$ and $\alpha_\nu'(t)$ on

$$\mathcal{O}_2 = \{(\nu, t) : 0 \le \nu \le 2 \ \text{ and } \ 2 \le t \le 1000\}. \tag{139}$$

A third set of functions B_1 and B_2 allow for the evaluation of $-\nu + \log(J_\nu(t)\sqrt{t})$ and $\nu + \log(-Y_\nu(t)\sqrt{t})$ on the subset

$$\mathcal{N}_1 = \left\{(\nu, t) : \nu \ge 2 \ \text{ and } \ \frac{\nu}{1000} < t < \sqrt{\nu^2 - \frac{1}{4}}\right\} \tag{140}$$

j	ξ_j	ξ_{j+1}	j	ξ_j	ξ_{j+1}
0	$\frac{1}{1{,}000{,}000{,}000}$	$\frac{1}{100{,}000{,}000}$	5	$\frac{1}{10{,}000}$	$\frac{1}{1{,}000}$
1	$\frac{1}{100{,}000{,}000}$	$\frac{1}{10{,}000{,}000}$	6	$\frac{1}{1{,}000}$	$\frac{1}{100}$
2	$\frac{1}{10{,}000{,}000}$	$\frac{1}{1{,}000{,}000}$	7	$\frac{1}{100}$	$\frac{1}{50}$
3	$\frac{1}{1{,}000{,}000}$	$\frac{1}{100{,}000}$	8	$\frac{1}{50}$	$\frac{1}{10}$
4	$\frac{1}{100{,}000}$	$\frac{1}{10{,}000}$	9	$\frac{1}{10}$	$\frac{1}{2}$

Table 1: The endpoints of the intervals $[\xi_j, \xi_{j+1}]$ used in Stage one of the procedure of Section 5.

of the nonoscillatory region $\mathcal{N}$. When ν is large, it is numerically advantageous to expand α_ν, α_ν', $-\nu + \log(J_\nu(t)\sqrt{t})$ and $\nu + \log(-Y_\nu(t)\sqrt{t})$ in powers of $\frac{1}{\nu}$ rather than in powers of ν. Consequently, in this procedure the functions A_1, C_1, B_1 and B_2 depend on $x = \frac{1}{\nu}$. Here, we only describe the construction of the functions A_1, C_1, B_1 and B_2. The procedure for the construction of A_2 and C_2 is quite similar, however.

There computations were conducted using IEEE extended precision arithmetic in order to ensure high accuracy. The resulting table, which consists of the coefficients in the expansions of A_1, C_1, A_2, C_2, B_1 and B_2, is approximately 1.3 megabytes in size. It allows for the evaluation of α_ν, α_ν', $-\nu + \log(J_\nu(t)\sqrt{t})$ and $\nu + \log(-Y_\nu(t)\sqrt{t})$ with roughly double precision relative accuracy (see the experiments of Section 7). The code was written in Fortran using OpenMP extensions and compiled with version 4.8.4 of the GNU Fortran compiler. It was executed on a computer equipped with 28 Intel Xeon E5-2697 processor cores running at 2.6 GHz. The construction of this table took approximately 227 seconds on this machine.

We conducted these calculations using extended precision arithmetic in order to ensure that the resulting expansions obtained full double precision accuracy. When these calculations are conducted in IEEE double precision arithmetic instead, only a small amount of precision is lost. We found that a table which can evaluate α_ν, α_ν', $-\nu + \log(J_\nu(t)\sqrt{t})$ and $\nu + \log(-Y_\nu(t)\sqrt{t})$ with roughly 12 digits of relative accuracy could be constructed using double precision arithmetic. Less than 5 seconds were required to do so.

Stage one: construction of the phase functions and logarithms

We began this stage of the procedure by constructing a partition

$$\xi_0 < \xi_1 < \xi_2 < \ldots < \xi_{10} \tag{141}$$

of the interval

$$\left[\frac{1}{1,000,000,000}, \frac{1}{2}\right]. \tag{142}$$

This partition divides (142) into ten subintervals, the endpoints of which are given in Table 1. For each such interval $[\xi_j, \xi_{j+1}]$, we formed the nodes

$$x_1^{(j)}, \ldots, x_{50}^{(j)} \tag{143}$$

of the 50-point Chebyshev grid on $[\xi_j, \xi_{j+1}]$. Next, for each x in the collection

$$x_1^{(0)}, \ldots, x_{50}^{(0)}, x_1^{(1)}, \ldots, x_{50}^{(1)}, \ldots, x_1^{(9)}, \ldots, x_{50}^{(9)} \tag{144}$$

we executed the algorithm of Section 4 with ν take to be $\frac{1}{x}$. The requested precision for the solver of Section 3 used by the algorithm of Section 4 was set to $\epsilon = 10^{-20}$ and we set the parameter n to be 50 so that the functions produced by the algorithm of Section 3 were represented via their values on the 50-point Chebyshev grids on a collection of subintervals. Were it not for the fact that the solver of Section 3 runs in time independent of ν, these calculations would have been prohibitively expensive to carry out, even on a massively parallel computer.

For each value of ν corresponding to one of the points (144), this resulted in the values of α_ν and α_ν' at the nodes of the 50-point Chebyshev grids on a collection of subintervals of

$$\left[\sqrt{\nu^2 - \frac{1}{4}}, 1000\nu\right) \tag{145}$$

and likewise for $-\nu + \log(J_\nu(t)\sqrt{t})$ and $\nu + \log(-Y_\nu(t)\sqrt{t})$ on a collection of subintervals of

$$\left[\frac{\nu}{1000}, \sqrt{\nu^2 - \frac{1}{4}}\right]. \tag{146}$$

As discussed in Section 2.8, this data allows for the evaluation of α_ν and its derivative at any point in (145), as well as the evaluation of $-\nu+\log(J_\nu(t)\sqrt{t})$ and $\nu+\log(-Y_\nu(t)\sqrt{t})$ at any point in (146).

Stage two: formation of unified discretizations

For each x in the set (144) we adaptively discretized the function $f_x : [0,1] \to \mathbb{R}$ defined via

$$f_x(y) = \alpha_\nu \left(\sqrt{\nu^2 - \frac{1}{4}} + \left(1000\nu - \sqrt{\nu^2 - \frac{1}{4}}\right) y \right) \quad \text{with} \quad \nu = \frac{1}{x} \tag{147}$$

using the procedure of Section 2.10. We requested $\epsilon = 10^{-17}$ precision and took the parameter n to be 49. Each discretization consisted of a collection of subintervals of $[0,1]$ on which f_x was represented to high accuracy using a 49-term Chebyshev expansion. We then formed a unified discretization

$$[a_0, a_1], [a_1, a_2], \ldots, [a_{24}, a_{25}] \tag{148}$$

of $[0,1]$ by merging these discretizations; that is, by ensuring that the sets (148) had the property that each subset appearing in the discretization of one of the functions f_x is the union of some collection of the subintervals (148).

For each x, we also adaptively discretized each of the functions $g_x : [0,1] \to \mathbb{R}$ and $h_x : [0,1] \to \mathbb{R}$ defined via the formulas

$$g_x(y) = -\nu + \log\left(J_\nu(t)\sqrt{t}\right) \quad \text{with} \quad \nu = \frac{1}{x},\ t = \frac{\nu}{1000} + \left(\sqrt{\nu^2 - \frac{1}{4}} - \frac{\nu}{1000}\right) y \tag{149}$$

and

$$h_x(y) = \nu + \log\left(-Y_\nu(t)\sqrt{t}\right) \quad \text{with} \quad \nu = \frac{1}{x},\ t = \frac{\nu}{1000} + \left(\sqrt{\nu^2 - \frac{1}{4}} - \frac{\nu}{1000}\right) y. \tag{150}$$

Again, we used the procedure of Section 2.10 with $\epsilon = 10^{-17}$ and $n = 49$. We then formed the unified discretization

$$[b_0, b_1], [b_1, b_2], \ldots, [b_{22}, b_{23}] \tag{151}$$

of $[0, 1]$ in the same fashion in which we formed (148).

Stage three: construction of the functions A_1 and C_1

The function A_1 is defined via the formula

$$A_1(x, y) = \frac{1}{\nu}\alpha_\nu\left(\sqrt{\nu^2 - \frac{1}{4}} + \left(1000\nu - \sqrt{\nu^2 - \frac{1}{4}}\right)y\right) \quad \text{with} \quad \nu = \frac{1}{x} \tag{152}$$

and C_1 is given by

$$C_1(x, y) = \frac{1}{\nu}\alpha'_\nu\left(\sqrt{\nu^2 - \frac{1}{4}} + \left(1000\nu - \sqrt{\nu^2 - \frac{1}{4}}\right)t\right) \quad \text{with} \quad \nu = \frac{1}{x}. \tag{153}$$

Obviously, A_1 and C_1 are defined on the compact rectangle

$$\left[\frac{1}{1,000,000,000}, \frac{1}{2}\right] \times [0, 1]. \tag{154}$$

For each $i = 0, \ldots, 9$ and each $j = 0, \ldots, 24$, we formed the 49^{th} order compressed bivariate Chebyshev expansions of A_1 and C_1 on the compact rectangle

$$[\xi_i, \xi_{i+1}] \times [a_j, a_{j+1}]. \tag{155}$$

There are 250 such rectangles and the uncompressed bivariate Chebyshev expansions of order 49 on each rectangle would involve $2,500$ coefficients. A total of $250 \times 2,500 = 625,000$ coefficients would be required to store the uncompressed bivariate expansions for A_1, and another $625,000$ would be required for C_1. The compressed bivariate expansions are much smaller. A mere $31,884$ values (this includes both the coefficients and the indices appearing in the sums (93), which must also be stored) were required to represent A_1. Only $51,076$ values were needed to represent C_1.

Stage four: construction of the functions B_1 and B_2

The function B_1 is defined via the formula

$$B_1(x, y) = -1 + \frac{1}{\nu}\log\left(J_\nu(t)\sqrt{t}\right) \quad \text{with} \quad \nu = \frac{1}{x},\ t = \frac{\nu}{1000} + \left(\sqrt{\nu^2 - \frac{1}{4}} - \frac{\nu}{1000}\right)y \tag{156}$$

and B_2 is defined by

$$B_1(x, y) = 1 + \frac{1}{\nu}\log\left(-Y_\nu(t)\sqrt{t}\right) \quad \text{with} \quad \nu = \frac{1}{x},\ t = \frac{\nu}{1000} + \left(\sqrt{\nu^2 - \frac{1}{4}} - \frac{\nu}{1000}\right)y. \tag{157}$$

Obviously, B_1 and B_2 are given on the compact rectangle

$$\left[\frac{1}{1,000,000,000}, \frac{1}{2}\right] \times [0, 1]. \tag{158}$$

For each $i = 0, \ldots, 9$ and each $j = 0, \ldots, 23$, we formed the 49^{th} order compressed bivariate

Chebyshev expansions of B_1 and B_2 on the compact rectangle

$$[\xi_i, \xi_{i+1}] \times [b_j, b_{j+1}] \,. \tag{159}$$

There are 230 such rectangles and the uncompressed bivariate Chebyshev expansions of B_1 and B_2 would involve $2 \times 230 \times 2,500 = 1,150,000$ coefficients. Using the compressed expansions, we are able to store B_1 using $32,910$ values and B_2 with $46,950$ values.

Remark 3. *It is possible to compute the values of both α_ν and α_ν' using the function A_1 via spectral differentiation (as discussed, for instance, [21]). Such an approach would, however, lead a level of loss of precision in the obtained values of α_ν' which we find unacceptable.*

In a similar vein, spectral integration could be used to evaluate α_ν given the values of α_ν'. Spectral integration does not suffer from the same defect as spectral differentiation and such a calculation could be carried out with little loss of precision; however, integration of α_ν' can only determine α_ν up to a constant. The appropriate constant would have to be calculated or stored in some fashion. We chose the simpler, but possibly more expensive, procedure described in this paper over such approach.

In order to evaluate $\alpha_\nu(t)$ and $\alpha_\nu'(t)$ given the expansions of A_1 and C_1 constructed using the procedure describe above, we execute the following sequence of steps:

1. First, we let

   $$x = \frac{1}{\nu} \tag{160}$$

   and

   $$y = \frac{t - \sqrt{\nu^2 - \frac{1}{4}}}{\left(1000\nu - \sqrt{\nu^2 - \frac{1}{4}}\right)}. \tag{161}$$

2. Next, we next search through the intervals (144) in order to find the index i of the one containing x and through the intervals (148) for index j of the interval containing y.

3. Having discovered that $(x, y) \in [\xi_i, \xi_{i+1}] \times [a_j, a_{j+1}]$, we evaluate the compressed bivariate Chebyshev series expansion representing A_1 on this rectangle. We scale the result by ν in order to obtain the value of $\alpha_\nu(t)$. We then evaluate the compressed bivariate Chebyshev expansion representing C_1 on this rectangle. We scale the result by ν in order to obtain the value of $\alpha_\nu'(t)$.

A virtually identical procedure is used to evaluate $\log(J_\nu(t))$ and $\log(-Y_\nu(t))$ using the expansions of B_1 and B_2 stored in the table.

6. An algorithm for the rapid numerical evaluation of Bessel functions

In this section, we describe the operation of our code for evaluating the Bessel functions J_ν and Y_ν of nonnegative orders and positive arguments. It was written in Fortran and its interface to the user consists of two subroutines, one called `bessel_eval_init` and the other `bessel_eval`. The `bessel_eval_init` routine reads the precomputed table constructed via the procedure of Section 5 from the disk into memory. Once this has been accomplished, the bessel_eval can be called. It takes as input an order $\nu \geq 0$ and an argument $t > 0$. When (ν, t) is in the oscillatory region $\mathcal{O}$, it returns the values of $\alpha_\nu(t)$ and $\alpha'_\nu(t)$ as well as those of $J_\nu(t)$ and $Y_\nu(t)$. When (ν, t) is in the nonoscillatory region $\mathcal{N}$, it returns the values of $\log(J_\nu(t))$ and $\log(-Y_\nu(t))$ as well as those of $J_\nu(t)$ and $Y_\nu(t)$. Of course, when $t \ll \nu$, these latter values might not be representable via the IEEE double format arithmetic. In this event, 0 is returned for $J_\nu(t)$ and $-\infty$ for $Y_\nu(t)$.

The bessel_eval code is available from the GitHub repository at address

http://github.com/JamesCBremerJr/BesselEval,

and from the author's website at the address

http://www.math.ucdavis.edu/~bremer/code.html.

The bessel_eval code operates as follows:

1. When $\nu \geq 2$ and $\sqrt{\nu^2 - \frac{1}{4}} \leq t \leq 1{,}000\nu$, the precomputed expansions of A_1 and C_1 are used to evaluate the nonoscillatory phase function α_ν and its derivative α'_ν at the point t. Then, formulas
$$J_\nu(t) = \sqrt{\frac{\pi t}{2}} \frac{\cos(\alpha_\nu(t))}{\sqrt{|\alpha'_\nu(t)|}} \tag{162}$$
and
$$Y_\nu(t) = \sqrt{\frac{\pi t}{2}} \frac{\sin(\alpha_\nu(t))}{\sqrt{|\alpha'_\nu(t)|}} \tag{163}$$
are used to produce the values of $J_\nu(t)$ and $Y_\nu(t)$.

2. When $\nu \geq 2$ and $\frac{\nu}{1000} \leq t < \sqrt{\nu^2 - \frac{1}{4}}$, the precomputed expansions of B_1 and B_2 are used to evaluate $-\nu + \log(J_\nu(t)\sqrt{t})$ and $\nu + \log(Y_\nu(t)\sqrt{t})$. The values of $J_\nu(t)$ and $Y_\nu(t)$ are calculated in the obvious fashion. Note that it is the the values of $\log(J_\nu(t))$ and $\log(-Y_\nu(t))$ and not those of $-\nu + \log(J_\nu(t)\sqrt{t})$ and $\nu + \log(Y_\nu(t)\sqrt{t})$ that are returned by the bessel_eval routine.

3. When $\nu > 100$ and $t \leq \frac{\nu}{1000}$, Debye's expansions (19) and (20) are used to evaluate $\log(J_\nu(t))$ and $\log(-Y_\nu(t))$. The values of $J_\nu(t)$ and $Y_\nu(t)$ are computed as one would expect.

4. When $\nu \leq 100$ and $t \leq \frac{\nu}{1000}$, the series expansions (17) and (18) are used to produce the values of $\log(J_\nu(t))$ and $\log(-Y_\nu(t))$. The values of $J_\nu(t)$ and $Y_\nu(t)$ are then computed as one would expect.

5. When $\nu < 2$ and $2 \leq t \leq 1000$, the precomputed expansions of A_2 and C_2 are used to evaluate the nonoscillatory phase function α_ν and its derivative α'_ν at the point t. Then, formulas (162) and (163) are used to produce the values of $J_\nu(t)$ and $Y_\nu(t)$.

6. When $\nu < 2$, $t < 2$ and (ν, t) is in the oscillatory region, we use the series expansions (13) and (15) in in order to evaluate $J_\nu(t)$ and $Y_\nu(t)$. As discussed in Section 2.2, Chebyshev interpolation is used in the computation of Y_ν when ν is either an integer or close to one.

The value of α_ν' is evaluated via the formula (40), and that of α_ν is calculated via

$$\alpha_\nu(t) = \arctan\left(\frac{Y_\nu(t)}{J_\nu(t)}\right). \tag{164}$$

7. When $\nu < 2$, $t < 2$ and (ν, t) is in the nonoscillatory regime, the series expansions (16) and (17) are used to evaluate $\log(J_\nu(t))$ and $\log(-Y_\nu(t))$. As discussed in Section 2.2, Chebyshev interpolation is used in the computation of Y_ν when ν is either an integer or close to one. The values of $J_\nu(t)$ and $Y_\nu(t)$ are calculated in the obvious fashion. We use series expansions rather than Debye's expansion to evaluate $J_\nu(t)$ and $Y_\nu(t)$ in this case because Debye's expansions lose accuracy when ν is small.

7. Numerical Experiments

In this section, we describe the results of numerical experiments conducted to illustrate the performance of the bessel_eval subroutine. These experiments were carried out on a laptop computer equipped with an Intel Core i7-5600U processor running at 2.6 GHz and 16GB of memory. Our code was compiled with the GNU Fortran compiler version 5.2.1 using the "-O3" compiler optimization flag. The size of the precomputed table used in the experiments of this paper is roughly 1.3 megabytes and, on the laptop used in our experiments, took approximately 10^{-2} seconds to read into memory.

7.1. The accuracy with which α_ν' is evaluated in the oscillatory region

In these experiments, we measured the accuracy with which bessel_eval calculates values of α_ν' in the oscillatory region. We did so by comparison with highly accurate reference values computed using version 11 of Wolfram's Mathematica package.

In each experiment, we first constructed $10,000$ pairs (ν, t) by first choosing ν in a given range and then randomly selecting a value of t in the interval

$$\left(\sqrt{\nu^2 - 1/4}, 1000\ \nu\right).$$

Unless, that is, $\nu < 1/2$, in which case we selected a random value of t in the interval $(0, 1000)$ instead. For each pair (ν, t) obtained in this fashion, we calculated the relative error in the value of $\alpha_\nu'(t)$ returned by bessel_eval. Table 2 displays the results. There, each row corresponds to one experiment and reports the maximum observed relative error as well as the average running time of bessel_eval.

7.2. The accuracy with which $-\nu + \log(J_\nu(t))$ and $-\nu + \log(-Y_\nu(t))$ are evaluated for small to moderate values of ν

In these experiments, we measured the accuracy with which bessel_eval calculates $-\nu+\log(J_\nu(t))$ and $-\nu + \log(-Y_\nu(t))$ in the nonoscillatory region. Reference values for these experiments were generated using version 11.0 of Wolfram's Mathematica package. A considerable amount of time is required for Mathematica to evaluate the Bessel functions $J_\nu(t)$ and $Y_\nu(t)$ when the magnitude of ν is large and t is small relative to ν. Consequently, in these experiments we only considered values of ν between $\frac{1}{2}$ and $10,000$. Larger values of ν were treated in the experiments described in the following section.

Range of ν	Maximum relative error in $\alpha'_\nu(t)$	Average evaluation time (in seconds)
0 - 1	4.44×10^{-16}	6.02×10^{-07}
1 - 10	1.11×10^{-16}	1.04×10^{-07}
10 - 100	1.11×10^{-16}	1.93×10^{-07}
100 - 1,000	1.11×10^{-16}	1.97×10^{-07}
1,000 - 10,000	1.11×10^{-16}	1.15×10^{-07}
10,000 - 100,000	1.11×10^{-16}	1.03×10^{-07}
100,000 - 1,000,000	1.11×10^{-16}	9.70×10^{-08}
1,000,000 - 10,000,000	3.33×10^{-16}	1.42×10^{-07}
10,000,000 - 100,000,000	1.11×10^{-16}	1.02×10^{-07}
100,000,000 - 1,000,000,000	1.11×10^{-16}	1.78×10^{-07}

Table 2: The results of the experiments of Section 7.1 in which the accuracy of the evaluation of α'_ν in the oscillatory region is tested through comparison with highly accurate reference values.

Range of ν	Maximum relative error in $-\nu + \log(J_\nu(t))$	Maximum relative error in $\nu + \log(-Y_\nu(t))$	Average evaluation time (in seconds)
0.5 - 1	2.43×10^{-16}	1.30×10^{-15}	2.11×10^{-06}
1 - 10	5.88×10^{-16}	8.48×10^{-16}	1.18×10^{-06}
10 - 100	7.06×10^{-16}	8.38×10^{-16}	8.05×10^{-07}
100 - 1,000	5.12×10^{-16}	7.57×10^{-16}	7.46×10^{-07}
1,000 - 10,000	6.41×10^{-16}	4.56×10^{-16}	5.70×10^{-07}

Table 3: The results of the experiments of Section 7.2 in which the accuracy of `bessel_eval` in the nonoscillatory region is tested via comparison with highly accurate reference values generated using Wolfram's Mathematica package.

In each experiment, we constructed $10,000$ pairs (ν, t) by first choosing a value of ν in a given range and then selecting a random point t in the interval

$$\left(0, \sqrt{\nu^2 - 1/4}\right).$$

For each pair (ν, t) obtained in this fashion, we calculated the relative accuracy of the quantities $-\nu + \log(J_\nu(t))$ and $\nu + \log(-Y_\nu(t))$. Table 3 displays the results of these experiments. There each row corresponds to one experiment and reports the largest relative errors which were observed as well as the average time taken by the `bessel_eval` routine.

7.3. The accuracy of the evaluation of $-\nu+\log(J_\nu(t))$ and $\nu+\log(-Y_\nu(t))$ deep in the nonoscillatory region

The `bessel_eval` subroutine makes use of the asymptotic expansions (19) and (20) when $0 < t < \nu/10,000$. For large ν, Debye expansion's are accurate in a much larger interval. In these experiments, we exploit this fact in order to measure the accuracy with which `bessel_eval` calculates $-\nu + \log(J_\nu(t))$ and $\nu + \log(-Y_\nu(t))$ deep in the nonoscillatory region.

In each experiment, we constructed $10,000$ pairs by first selecting a value of ν in a given range at random and then picking a random value of t in the interval $(\nu/1000, \nu/10)$. For each pair (ν, t) obtained in this fashion, we computed the values of both $-\nu + \log(J_\nu(t))$ and $\nu + \log(-Y_\nu(t))$ using `bessel_eval` and compared them to reference values obtained using Debye's expansion. The reference calculations were performed using IEEE quadruple precision arithmetic in order to ensure high accuracy. The results are shown in Table 4. Each row there corresponds to one experiment and reports the range of ν, the maximum relative error which was observed, and the average time taken by `bessel_eval`.

Range of ν	Maximum relative error in $-\nu + \log(J_\nu(t))$	Maximum relative error in $\nu + \log(-Y_\nu(t))$	Average evaluation time (in seconds)
100 - 1,000	8.26×10^{-16}	7.99×10^{-16}	4.50×10^{-07}
1,000 - 10,000	8.88×10^{-16}	9.00×10^{-16}	3.97×10^{-07}
10,000 - 100,000	9.13×10^{-16}	8.52×10^{-16}	3.84×10^{-07}
100,000 - 1,000,000	7.62×10^{-16}	8.71×10^{-16}	4.15×10^{-07}
1,000,000 - 10,000,000	7.45×10^{-15}	7.39×10^{-15}	3.62×10^{-07}
10,000,000 - 100,000,000	8.62×10^{-16}	7.66×10^{-16}	3.59×10^{-07}
100,000,000 - 1,000,000,000	7.49×10^{-16}	9.38×10^{-16}	3.87×10^{-07}

Table 4: The results of the experiments of Section 7.3 in which the accuracy with which $-\nu + \log(J_\nu(t))$ and $\nu + \log(-Y_\nu(t))$ is tested for values of $100 \leq \nu \leq 1,000,000,000$ and $t \ll \nu$ through comparison with Debye's expansions.

7.4. The accuracy of the evaluation of $J_\nu(t)$ and $Y_\nu(t)$ as a function of t

In these experiments, we measured the relative accuracy with which `bessel_eval` calculates the Hankel function of the first kind $H_\nu(t) = J_\nu(t) + iY_\nu(t)$ as a function of t. We considered the Hankel function instead of treating J_ν and Y_ν separately because $H_\nu(t)$ does not vanish in the interval $(0, \infty)$ and its absolute value is nonoscillatory there, properties not shared by the Bessel functions J_ν and Y_ν.

In each experiment, we chose a value of ν and measured the relative accuracy with which `bessel_eval` calculates $J_\nu(t) + iY_\nu(t)$ at each of $1,000$ equispaced points in the interval $[\nu, 100000\nu]$. Highly accurate reference values for these experiments were computed using Mathematica. We chose the following values of ν: $\sqrt{2}$, $10\sqrt{2}$, $100\sqrt{2}$ and $1,000\sqrt{2}$. We also repeated these experiments using Amos' well-known code [1].

Figure 2 displays the results. Each graph there plots the base-10 logarithms of the relative errors in the calculated values of $J_\nu(t)+iY_\nu(t)$ as dots. The graph of the function $\kappa(t)\epsilon_0$, where $\kappa(t)$ is the condition number of the evaluation of H_ν at the point t and $\epsilon_0 = 2^{-52} \approx 2.22044604925031 \times 10^{-16}$ is machine epsilon, is also plotted as a solid curve. The results for `bessel_eval` are shown on the left while those for Amos' code appear on the right on the right.

7.5. The speed and accuracy of the evaluation of J_n and Y_n as a function of n

In these experiments, we compared the speed and accuracy with which `bessel_eval` calculates Hankel functions of integer orders with the speed and accuracy of Amos' code [1]. Reference values were calculated using the well-known three-term recurrence relations satisfied by Bessel functions. The calculation of reference values was performed using IEEE quadruple precision arithmetic in order to ensure high accuracy.

	`bessel_eval`		`Amos' code`	
n	Maximum relative error in H_n	Average evaluation time (in seconds)	Maximum relative error in H_n	Average evaluation time (in seconds)
0	3.02×10^{-13}	4.38×10^{-07}	9.32×10^{-15}	6.00×10^{-07}
1	3.08×10^{-13}	2.68×10^{-07}	5.03×10^{-14}	6.64×10^{-07}
10	3.42×10^{-12}	2.86×10^{-07}	9.04×10^{-13}	1.40×10^{-06}
100	3.36×10^{-11}	2.47×10^{-07}	1.78×10^{-11}	1.98×10^{-06}
1,000	2.45×10^{-10}	2.58×10^{-07}	1.90×10^{-10}	2.00×10^{-06}
10,000	3.38×10^{-09}	2.35×10^{-07}	2.62×10^{-09}	1.80×10^{-06}
100,000	3.21×10^{-08}	2.26×10^{-07}	1.84×10^{-08}	1.67×10^{-06}
1,000,000	2.93×10^{-07}	2.36×10^{-07}	-	-
10,000,000	2.67×10^{-06}	2.18×10^{-07}	-	-
100,000,000	2.97×10^{-05}	2.06×10^{-07}	-	-
1,000,000,000	2.83×10^{-04}	1.97×10^{-07}	-	-

Table 5: The results of the experiments of Section 7.5 in which the speed and accuracy with which `bessel_eval` and the well-known code of Amos [1] evaluates Hankel functions of integer orders is compared. Experiments in which Amos' code returned an error code are marked with dashes.

In the first experiment, n was taken to be 0 and $10,000$ random points at which to evaluate H_n were chosen in the interval $(0, 1000)$. In each subsequent experiment, n was taken to be a positive integer and $10,000$ random points at which to evaluate H_n were chosen from the interval

$$(a_n, 1000n)\,, \tag{165}$$

where $a_n < \sqrt{n^2 - 1/4}$ is the solution of the equation $\log(-Y_n(a)) = 100$. In this way, we avoided problems with numerical overflow and underflow. At each point chosen in this fashion, the value of the Hankel function H_n was calculated using `bessel_eval` and with Amos' code. Table 5 reports the results. There, the maximum observed relative error in the values of H_n generated by each code is reported as a function of n as is the average time taken by each code to perform an evaluation. Amos' code aborts and returns an error code in cases in which it is unable to evaluate the Bessel functions to at least 7-digit accuracy. The corresponding entries of Table 5 are marked with dashes.

7.6. The accuracy of the evaluation of $J_\nu(t)$ and $Y_\nu(t)$ in the nonoscillatory regime

The preceding section reports on the accuracy with which the algorithm of this paper evaluates J_ν and Y_ν. The experiments described there are global in nature; that is, we report the largest error which was observed in evaluating the Bessel functions on an interval including both the oscillatory and nonsocillatory regimes. In fact, less accuracy is lost when evaluating J_ν and Y_ν in the nonoscillatory regime. In this set of experiments, we measured the accuracy with which `bessel_Eval` evauates J_ν and Y_ν in the nonoscillatory regime only.

In each experiment, we constructed $10,000$ pairs (ν, t) by first choosing a value of ν in a given range and then selecting a random point t in the nonoscillatory interval for J_ν and Y_ν. For each pair (ν, t) obtained in this fashion, we calculated the relative accuracy of $J_\nu(t)$ and $Y_\nu(t)$. The results are shown in Table 6.

Range of ν	Maximum relative error in $J_\nu(t)$	Maximum relative error in $Y_\nu(t)$	Average evaluation time (in seconds)
100 - 1,000	9.85×10^{-13}	9.63×10^{-13}	8.31×10^{-07}
1,000 - 10,000	8.97×10^{-12}	7.67×10^{-12}	5.22×10^{-07}
10,000 - 100,000	4.81×10^{-11}	4.64×10^{-11}	4.37×10^{-07}
100,000 - 1,000,000	2.50×10^{-10}	4.10×10^{-10}	3.88×10^{-07}
1,000,000 - 10,000,000	1.89×10^{-10}	9.01×10^{-10}	3.84×10^{-07}
10,000,000 - 100,000,000	3.49×10^{-09}	1.21×10^{-09}	3.52×10^{-07}
100,000,000 - 1,000,000,000	1.50×10^{-07}	9.20×10^{-08}	3.63×10^{-07}

Table 6: The results of the experiments of Section 7.6 in which the accuracy with which `bessel_eval` calculates Bessel functions in the nonoscillatory regime is measured.

7.7. Extended precision experiments

It is a straightforward to increase the accuracy of the precomputed expansions used by the algorithm of this paper. We constructed a second set of these expansions, this time asking for 25 digits of accuracy. Of course, these precomputations were conducted using IEEE quadruple precision arithmetic. We then reran the experiments of Sections 7.1, 7.2, 7.3 and 7.5 using IEEE quadruple precision arithmetic instead of the standard IEEE double precision arithmetic. Because the laptop we used for experiments does not support quadruple precision arithmetic in hardware, it was emulated with software. This is, of course, highly inefficient and the running times of these experiments reflect this fact. The results are shown in Tables 7 through 10. Moreover, because Amos' code was not designed for extended precision arithmetic, we omitted comparion with it (it should be noted, though, that the mere fact that a quadruple precision code could be produced so easily is a significant advantage of our approach).

Range of ν	Maximum relative error in $\alpha'_\nu(t)$	Average evaluation time (in seconds)
0 - 1	2.05×10^{-28}	1.79×10^{-05}
1 - 10	3.47×10^{-28}	5.70×10^{-06}
10 - 100	9.62×10^{-35}	5.47×10^{-06}
100 - 1,000	9.02×10^{-29}	5.31×10^{-06}
1,000 - 10,000	1.71×10^{-32}	5.45×10^{-06}
10,000 - 100,000	7.60×10^{-33}	5.30×10^{-06}
100,000 - 1,000,000	9.62×10^{-34}	5.24×10^{-06}
1,000,000 - 10,000,000	9.62×10^{-35}	5.26×10^{-06}
10,000,000 - 100,000,000	9.62×10^{-35}	5.28×10^{-06}
100,000,000 - 1,000,000,000	2.48×10^{-28}	5.37×10^{-06}

Table 7: The results obtained by rerunning the experiments of Section 7.1 using IEEE quadruple precision arithmetic. These experiments measure the accuracy of the evaluation of α'_ν in the oscillatory region.

8. Conclusions and future work

Using a simple-minded procedure which can be applied to a large class of special functions with little modification, we constructed a table which allows for the numerical evaluation of Bessel functions

Range of ν	Maximum relative error in $-\nu+\log(J_\nu(t))$	Maximum relative error in $\nu+\log(-Y_\nu(t))$	Average evaluation time (in seconds)
0.5 - 1	3.83×10^{-34}	2.54×10^{-28}	4.65×10^{-05}
1 - 10	2.26×10^{-25}	1.18×10^{-27}	1.41×10^{-04}
10 - 100	7.63×10^{-25}	1.09×10^{-27}	9.97×10^{-05}
100 - 1,000	8.19×10^{-28}	1.61×10^{-27}	9.51×10^{-05}
1,000 - 10,000	4.57×10^{-28}	7.80×10^{-28}	4.93×10^{-05}

Table 8: The results of rerunning the experiments of Section 7.2 using IEEE quadruple precision arithmetic. These experiments test the accuracy of `bessel_eval` in the nonoscillatory region via comparison with highly accurate reference values generated using Wolfram's Mathematica package.

Range of ν	Maximum relative error in $-\nu+\log(J_\nu(t))$	Maximum relative error in $\nu+\log(-Y_\nu(t))$	Average evaluation time (in seconds)
100 - 1,000	3.00×10^{-28}	2.58×10^{-28}	3.53×10^{-05}
1,000 - 10,000	2.95×10^{-28}	2.55×10^{-28}	2.73×10^{-05}
10,000 - 100,000	2.47×10^{-28}	2.38×10^{-28}	2.28×10^{-05}
100,000 - 1,000,000	2.67×10^{-28}	2.27×10^{-28}	2.01×10^{-05}
1,000,000 - 10,000,000	2.22×10^{-28}	2.43×10^{-28}	1.83×10^{-05}
10,000,000 - 100,000,000	1.62×10^{-28}	1.69×10^{-28}	1.70×10^{-05}
100,000,000 - 1,000,000,000	2.66×10^{-28}	2.76×10^{-28}	1.59×10^{-05}

Table 9: The results of rerunning the experiments of Section 7.3 using IEEE quadruple precision arithmetic. These experiments test the accuracy with which $-\nu+\log(J_\nu(t))$ and $\nu+\log(-Y_\nu(t))$ is evaluated for values of $100\leq\nu\leq 1,000,000,000$ and $t\ll\nu$ through comparison with Debye's expansions.

n	Maximum relative error in H_n	Average evaluation time (in seconds)
0	3.00×10^{-26}	5.41×10^{-05}
1	2.17×10^{-25}	3.02×10^{-05}
10	1.85×10^{-24}	3.16×10^{-05}
100	2.39×10^{-23}	2.48×10^{-05}
1,000	2.45×10^{-22}	2.14×10^{-05}
10,000	8.01×10^{-22}	1.90×10^{-05}
100,000	1.48×10^{-20}	1.77×10^{-05}
1,000,000	6.08×10^{-20}	1.70×10^{-05}
10,000,000	8.52×10^{-19}	1.57×10^{-05}
100,000,000	7.62×10^{-18}	1.51×10^{-05}
1,000,000,000	5.57×10^{-17}	1.53×10^{-05}

Table 10: The results of rerunning the experiments of Section 7.7 using IEEE quadruple precision arithmetic. In these experiments, the speed and accuracy with which `bessel_eval` evaluates Hankel functions of integer orders is measured.

of nonnegative real orders and positive arguments. In the regime, the performance of the resulting code is comparable to that of the well-known and widely used code of Amos [1].

In the nonoscillatory region, our algorithm calculates the logarithms of the Bessel functions as well as their values. This is useful in cases in which the magnitudes of the Bessel functions themselves are too large or too small to be encoded using the IEEE double precision format. In the oscillatory region, in addition to the values of the Bessel function itself, our algorithm also returns the values of a nonoscillatory phase function for Bessel's equation and its derivative. This is extremely helpful when computing the zeros of special functions [3], and when applying special function transforms via the butterfly algorithm (see, for instance, [6, 19, 20, 7, 23, 26]).

Amos' code allows for the numerical evaluation of Bessel functions of complex arguments. The phase function approach can be extended to do so as well. Indeed, Bessel's differential equation admits a phase function which is nonoscillatory on the upper half of the complex plane. Moreover, that phase function is related to a solution of Bessel's differential equation which is an element of the Hardy space of functions analytic on the complex plane. These observations can be exploited in order to efficiently evaluate Bessel functions in the upper half of the complex plane; such a method will be reported by the author at a later date.

The author will report on the use of the method of this paper to evaluate associated Legendre functions and prolate spheroidal wave functions at a later date, as well as on the rapid application of special function transforms using techniques related to those discussed here.

9. Acknowledgments

The author was supported by National Science Foundation grant DMS-1418723, and by a UC Davis Chancellor's Fellowship.

10. References

References

[1] Amos, D. E. Algorithm 644: a portable package for Bessel functions of a complex argument and nonnegative order. *ACM Transactions on Mathematica Software 3* (1986), 265–273.

[2] Bochner, S., and Martin, W. *Several Complex Variables.* Princeton University Press, 1948.

[3] Bremer, J. On the numerical calculation of the roots of special functions satisfying second order ordinary differential equations. *SIAM Journal on Scientific Computing 39* (2017), A55–A82.

[4] Bremer, J. On the numerical solution of second order differential equations in the high-frequency regime. *Applied and Computational Harmonic Analysis* (2017), to appear.

[5] Bremer, J., and Rokhlin, V. Improved estimates for nonoscillatory phase functions. *Discrete and Continuous Dynamical Systems, Series A 36* (2016), 4101–4131.

[6] Candés, E., Demanet, L., and Ying, L. Fast computation of Fourier integral operators. *SIAM Journal on Scientific Computing* (2007), 2464–2493.

[7] Candés, E., Demanet, L., and Ying, L. Fast butterfly algorithm for the computation of Fourier integral operators. *SIAM Journal on Multiscale Modeling and Simulation* (2009), 1727–1750.

[8] Carleson, L. On convergence and growth of partial sums of Fourier series. *Acta Mathematica 116* (1966), 135–157.

[9] *NIST Digital Library of Mathematical Functions.* http://dlmf.nist.gov/, Release 1.0.13 of 2016-09-16. F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller and B. V. Saunders, eds.

[10] Erdélyi, A., et al. *Higher Transcendental Functions*, vol. II. McGraw-Hill, 1953.

[11] Fedoryuk, M. V. *Asymptotic Analysis.* Springer-Verlag, 1993.

[12] Fefferman, C. On the convergence of multiple Fourier series. *Bulletin of the American Mathematical Society 77* (1971), 744–745.

[13] Gil, A., Segura, J., and Temme, N. M. *Numerical Methods for Special Functions.* SIAM, 2007.

[14] Heitman, Z., Bremer, J., Rokhlin, V., and Vioreanu, B. On the asymptotics of Bessel functions in the Fresnel regime. *Applied and Computational Harmonic Analysis 39* (2015), 347–355.

[15] Higham, N. J. *Accuracy and Stability of Numerical Algorithms*, second ed. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2002.

[16] Hille, E. *Ordinary differential equations in the complex domain.* Wiley, New York, 1976.

[17] Iserles, A. *A First Course in the Numerical Analysis of Differential Equations.* Cambridge University Press, 1996.

[18] Kummer, E. De generali quadam aequatione differentiali tertti ordinis. *Progr. Evang. Köngil. Stadtgymnasium Liegnitz* (1834).

[19] Li, Y., and Yang, H. Interpolative butterfly factorization. *SIAM Journal on Scientific Computing*, to appear.

[20] Li, Y., Yang, H., Martin, E., Ho, K. L., and Ying, L. Butterfly factorization. *SIAM Journal on Multiscale Modeling and Simulation 13* (2015), 714–732.

[21] Mason, J., and Handscomb, D. *Chebyshev Polynomials.* Chapman and Hall, 2003.

[22] Matviyenko, G. On the evaluation of Bessel functions. *Applied and Computational Harmonic Analysis 1* (1993), 116–135.

[23] Michielssen, E., and Boag, A. A multilevel matrix decomposition algorithm for analyzing scattering from large structures. *IEEE Transactions Antennas and Propagation 44* (1996), 1086–1093.

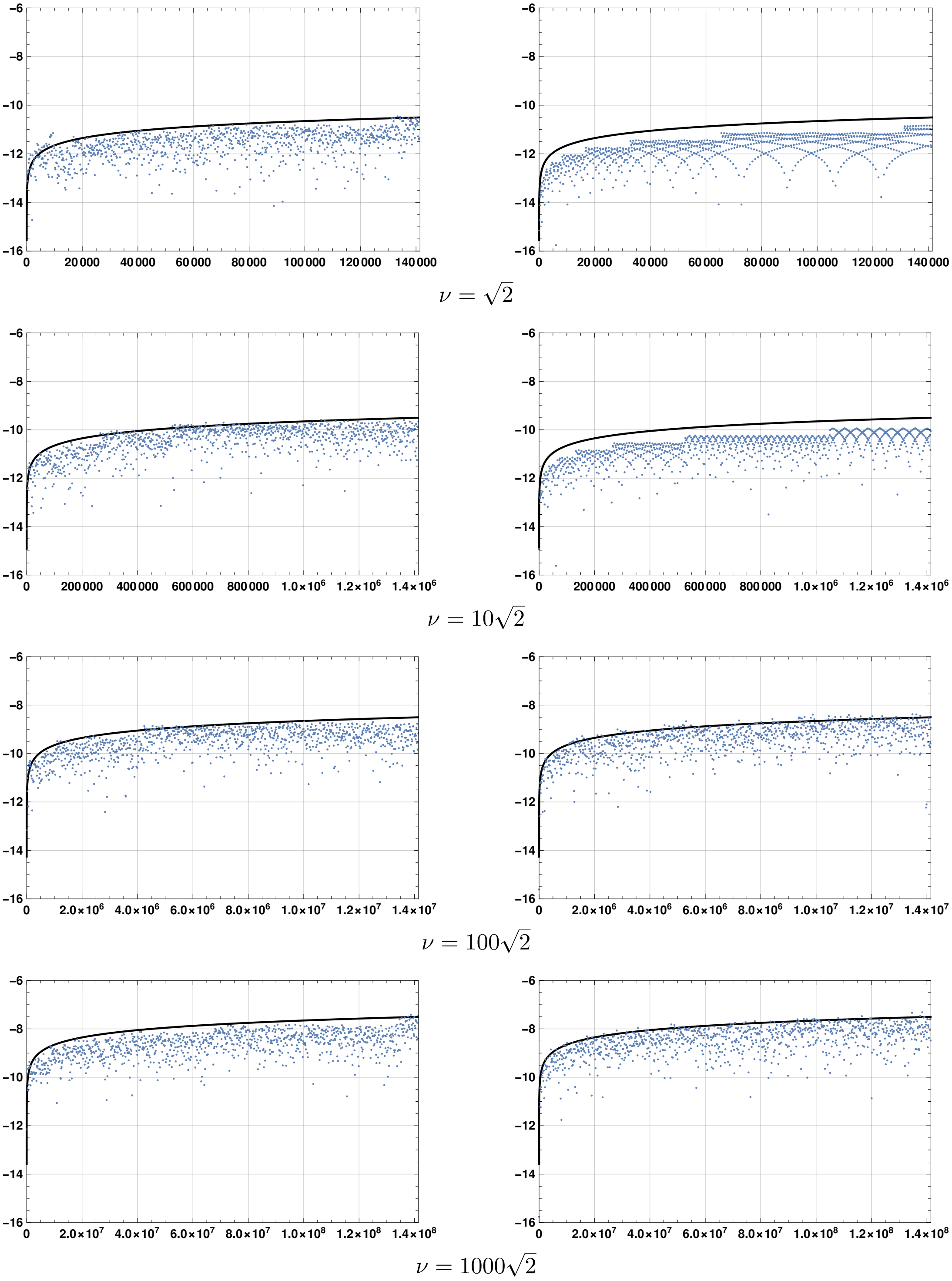
[24] Miller, J. On the choice of standard solutions for a homogeneous linear differential equation of the second order. *Quarterly Journal of Mechanics and Applied Mathematics 3* (1950), 225–235.

[25] Olver, F. W. *Asymptotics and Special Functions.* A.K. Peters, Natick, MA, 1997.

[26] O'Neil, M., Woolfe, F., and Rokhlin, V. An algorithm for the rapid evaluation of special function transforms. *Applied and Computational Harmonic Analysis 28* (2010), 203–226.

[27] Trefethen, N. *Approximation Theory and Approximation Practice.* Society for Industrial and Applied Mathematics, 2013.

[28] Watson, G. N. *A Treatise on the Theory of Bessel Functions*, second ed. Cambridge University Press, New York, 1995.



$\nu = \sqrt{2}$

$\nu = 10\sqrt{2}$

$\nu = 100\sqrt{2}$

$\nu = 1000\sqrt{2}$

Figure 2: The results of the experiments of Section 7.4. In each graph, the base-10 logarithm of the relative errors in calculated values of $H_\nu(t)$ are plotted as dots and the graph of the function $\log_{10}(\kappa(t))\epsilon_0$, where $\kappa(t)$ is the condition number of the evaluation of the function $H_\nu(t)$ and ϵ_0 is machine epsilon, is plotted as a solid line. The plots on the left show the results obtained using the `bessel_eval` routine while those on the right show results obtained from Amos' well-known and widely used code [1].